Queueing Analysis of a Priority-based Claim Processing System

by
Basil Karim Wagih Ibrahim

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Actuarial Science

Waterloo, Ontario, Canada, 2009

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We propose a situation in which a single employee is responsible for processing incoming claims to an insurance company that can be classified as being one of two possible types. More specifically, we consider a priority-based system having separate buffers to store high priority and low priority incoming claims. We construct a mathematical model and perform queueing analysis to evaluate the performance of this priority-based system, which incorporates the possibility of claims being redistributed, lost, or prematurely processed.

Acknowledgements

I would like to thank my supervisor, Professor Steve Drekic, for introducing me to this work and helping derive and calculate the mathematical elements of the model used here. You have been a great inspiration and I am looking forward to continuing my education under your supervision. I would also like to thank Professors Jun Cai and Gordon Willmot for agreeing to read my thesis. Your feedback will help me make this work as readable and mathematically accurate as possible.

Contents

List of Tables

List of Figures

Chapter 1

Preliminaries

1.1 Introduction

With the increasing number of insurance policies issued every year, especially in first world countries, it has become essential for insurance companies – alongside with implementing effective marketing strategies and setting market competitive premium prices and claim amounts – to use effective and efficient systems for registering and processing claims filed by their insurance policyholders. Insurance companies that do not process and settle the claims of their customers efficiently and reliably face losing their credibility and competitiveness in the market, which can eventually lead to potential bankruptcy. In other words, establishing and maintaining optimally efficient claim processing systems should be one of the essential agendas of insurance companies, in order to remain successful.

This thesis assumes a particular queueing scenario of claims entering an employee's data-system, waiting to be processed. By making certain mathematical assumptions, we examine important probability distributions that would help employees/insurance companies in optimizing her/his/its claims management system(s). On a bigger scale, we believe that using mathematically tractable models, formulated by realistic assumptions, is a critical tool for optimizing the performance of organizations – especially for those that are exposed to high levels of risk.

1.2 The Model

We assume that the claims registration system for employees in a particular insurance company prioritizes its incoming claims into two first-come, first-serve-basis queues: a high-priority (HP for short) queue and a low-priority (LP for short) queue. If an employee is processing an HP claim, she/he will not attend to an LP claim until the entire HP queue is empty. However, if the employee is currently processing an LP claim, she/he will attend to claims in the HP queue – provided that it is not empty – after having finished the current LP claim. Otherwise, she/he will move on to process the next LP claim. This is referred to as a non-preemptive service rule.

Finally, the model incorporates the possibility that claims, while waiting in the system to be processed, can be removed from the queue. This can be due to reasons such as premature processing, loss or redistribution (to another queue, another department, etc.). For instance, it can happen there are claims that may need to be reallocated to another system, or may require more time or expertise to be processed. This will help minimize delays in processing the remaining claims in the system, thereby making it more efficient and dependable. To simplify our description of the model, and without loss of generality, we will label premature queue exiting (i.e. reneging) as "redistribution". Hence, by incorporating this option, our model provides a higher degree of flexibility than many classical queueing models whose elements remain in the system until service.

These are the two main objectives of this thesis:

1. Derive the waiting-time distribution of an arbitrarily arriving HP claim.
2. Derive the waiting-time distribution of an arbitrarily arriving LP claim.

Before proceeding to the mathematical aspects of this model, we provide a couple of examples in which such a priority-based system could possibly be implemented in practice:

1. Claims that are easy and quick to process (e.g. dental insurance claims, medical insurance) are assigned as HP. Claims that are more difficult and time consuming to assess (e.g. property damage, moral hazard) are assigned as LP.

2. Claims of high amounts and requiring immediate settlement (e.g. damage of infrastructure due to a natural disaster) are assigned as HP. Claims of a less important nature (e.g. tooth damage of a celebrity) are assigned as LP.

Therefore, the model proposed here is not restricted to a particular class of insurance policy claims, providing flexibility for a variety of practical applications. We now proceed to the mathematical details of the model.

1.3 Mathematical Notation

The following figure provides a visual depiction of what our proposed queueing model looks like:

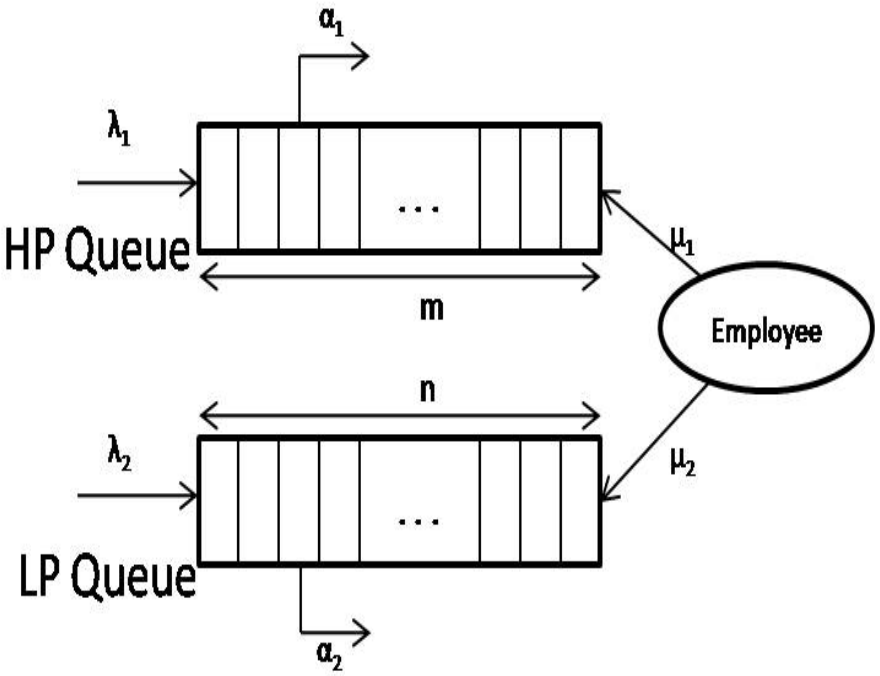

Figure 1.1: Proposed Queueing Model

The following table shows the main notation that will be used throughout the thesis. It is also worth mentioning that the notation used in the subsequent chapters, other than those contained in Table 1.1, should be treated as local (i.e. a letter used to represent some value or expression in one chapter can be used differently in another chapter).

Notation	Description
m	Maximum finite capacity of the HP queue
n	Maximum finite capacity of the LP queue
λ_1	Entrance rate into the HP queue
λ_2	Entrance rate into the LP queue
μ_1	Processing or service rate for an HP claim ($1/\mu_1$ is the mean HP service time)
μ_2	Processing or service rate for an LP claim ($1/\mu_2$ is the mean LP service time)
α_1	Redistribution rate of an HP claim
α_2	Redistribution rate of an LP claim
X_1	Steady-state number of HP claims in the queue, including the one being processed
X_2	Steady-state number of LP claims in the queue, including the one being processed
P	Type of claim the employee is currently processing (takes values of H and L)
$\pi_{i,j,H}$	$Pr\{X_1 = i, X_2 = j, P = H\}$
$\pi_{i,j,L}$	$Pr\{X_1 = i, X_2 = j, P = L\}$
$\pi_{i,j}$	$Pr\{X_1 = i, X_2 = j\} = \pi_{i,j,H} + \pi_{i,j,L}$
W^H	Waiting time of an arbitrarily arriving HP claim, assuming it cannot be redistributed
W^L	Waiting time of an arbitrarily arriving LP claim, assuming it cannot be redistributed
$W^H_{i,j}$	W^H, given i HP claims, j LP claims
$W^H_{i,j,H}$	W^H, given i HP claims, j LP claims, employee currently processing an HP claim
$W^H_{i,j,L}$	W^H, given i HP claims, j LP claims, employee currently processing an LP claim
R^H	Redistribution time of an HP claim
R^L	Redistribution time of an LP claim
T^H	Time spent in the HP queue for an arbitrarily arriving HP claim
$T^H\|NotF$	Time spent in the HP queue, excluding possibility that the HP queue is full
T^L	Time spent in the LP queue for an arbitrarily arriving LP claim
$T^L\|NotF$	Time spent in the LP queue, excluding possibility that the LP queue is full
I_i	Identity matrix of dimension i-by-i
T'	Transpose of a matrix T
$X \sim Exp(\beta)$	Random variable X is exponentially distributed with mean $1/\beta$
$X \sim PH_i(\underline{\alpha}, T)$	Random variable X is phase-type distributed with initial probability row-vector $\underline{\alpha}$ and generator matrix T of dimension i
$exp\{T\}$	Matrix exponential of a square matrix T, namely $\sum_{i=0}^{\infty} \frac{T^i}{i!}$
$F_X(\omega)$	Cumulative distribution function (cdf) of a random variable X
$E[X]$	Expectation of a random variable X
$Var(X)$	Variance of a random variable X

Table 1.1: Notation for Our Queueing Model

1.4 Model Assumptions

In order to make the model mathematically tractable, it is necessary to make assumptions that make explicit mathematical calculations possible. The main challenge in this is to be able to strike a right balance between having a model in which it is possible to derive/calculate/simulate all its variables and a model that is realistic. As it is usually the case that there is a tradeoff between the two options, one has to find the best compromise.

In this case, due to the context for which this model is proposed, one may consider the assumptions we are about to make plausible. For future research, as we will mention in the last chapter, one may relax some of these assumptions and attempt to derive/calculate the underlying mathematical expressions/quantities of a new, more general model.

The following is a list of the assumptions made in our model:

1. The employee only handles one claim at a time.
2. Incoming claims face four possibilities: enter (if possible) a queue, wait in the queue, get redistributed or proceed to service.
3. Claims receiving service are no longer subject to redistribution.
4. All rates are independent and exponentially distributed.

With these assumptions, one obtains a continuous-time Markov process. For details on continuous-time Markov processes, we refer the reader to Chapter 6 of [3]. We now turn our attention to the next chapter, which summarizes the main results of [1], in which the form of the infinitesimal generator matrix is specified as well as an algorithm to compute the associated steady-state probabilities $\pi_{i,j}$'s.

Chapter 2

Calculating the Steady-State Probabilities

As the main contribution of [1] involved the computation of $\{\pi_{i,j,P} : i = 0, 1, 2, \ldots, m;\ j = 0, 1, 2, \ldots, n; P = H, L\}$, we simply summarize the main findings here, and then use these results for calculating the waiting-time distributions whose derivations are shown in the subsequent chapters.

Define:

- $\underline{\pi} = (\underline{\pi}_0, \underline{\pi}_1, \ldots, \underline{\pi}_m)$.
- $\underline{\pi}_0 = (\pi_{0,0},\ \pi_{0,1},\ \ldots,\ \pi_{0,n})$.
- For $k = 1, 2, ..., m$,
  $\underline{\pi}_k = (\pi_{k,0},\ \pi_{k,1,H},\ \pi_{k,1,L},\ \ldots,\ \pi_{k,n,H},\ \pi_{k,n,L})$.

The main objective becomes solving the equilibrium equations given by $\underline{\pi}Q = \underline{\tilde{\mathbf{0}}}$, where Q is the infinitesimal generator matrix of the process and $\underline{\tilde{\mathbf{0}}}$ is a row vector of zeroes with the same dimension as Q. In [1], the form of Q is first derived followed by a computer-efficient methodology for calculating the steady-state probabilities $\pi_{i,j,P}$'s.

2.1 Calculating the Infinitesimal Generator Matrix Q

By analyzing the simple, albeit tedious, transition states of the model, one finds that Q is of dimension $(mn + (m + 1)(n + 1))$-by-$(mn + (m + 1)(n + 1))$, exhibiting the following block-structured form:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ m-2 \\ m-1 \\ m \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & \cdots & m-2 & m-1 & m \end{array} \\ \begin{pmatrix} Q_{0,0} & Q_{0,1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ Q_{1,0} & Q_{1,1} & Q_{1,2} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q_{2,1} & Q_{2,2} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & Q_{m-2,m-2} & Q_{m-2,m-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & Q_{m-1,m-2} & Q_{m-1,m-1} & Q_{m-1,m} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & Q_{m,m-1} & Q_{m,m} \end{pmatrix} \end{array},$$

where:

- $Q_{0,0}$ is $(n + 1)$-by-$(n + 1)$.
- $Q_{0,1}$ is $(n + 1)$-by-$(2n + 1)$.
- $Q_{1,0}$ is $(2n + 1)$-by-$(n + 1)$.
- All other submatrices are of size $(2n + 1)$-by-$(2n + 1)$.

Since all rates are independent and exponentially distributed, one can derive explicitly the elements of all these submatrices. Extracted from [1], we have:

$$Q_{0,0} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & \cdots & n-1 & n \end{array} \\ \begin{pmatrix} -\lambda & \lambda_2 & 0 & \cdots & 0 & 0 \\ \gamma_0 & -(\lambda+\gamma_0) & \lambda_2 & \ddots & 0 & 0 \\ 0 & \gamma_1 & -(\lambda+\gamma_1) & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -(\lambda+\gamma_{n-2}) & \lambda_2 \\ 0 & 0 & 0 & \cdots & \gamma_{n-1} & -(\lambda_1+\gamma_{n-1}) \end{pmatrix} \end{array},$$

where $\lambda = \lambda_1 + \lambda_2$ and $\gamma_i = \mu_2 + i\alpha_2$.

We also have:

$$Q_{0,1} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1_H & 1_L & 2_H & 2_L & \cdots & n_H \quad n_L \end{array} \\ \begin{pmatrix} \lambda_1 & \begin{bmatrix} 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \end{bmatrix} & \cdots & \begin{bmatrix} 0 & 0 \end{bmatrix} \\ 0 & \begin{bmatrix} 0 & \lambda_1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \end{bmatrix} & \cdots & \begin{bmatrix} 0 & 0 \end{bmatrix} \\ 0 & \begin{bmatrix} 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & \lambda_1 \end{bmatrix} & \cdots & \begin{bmatrix} 0 & 0 \end{bmatrix} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \begin{bmatrix} 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \end{bmatrix} & \cdots & \begin{bmatrix} 0 & \lambda_1 \end{bmatrix} \end{pmatrix} \end{array}, \text{ and}$$

$$Q_{1,0} = \begin{array}{c} \\ 0 \\ 1_H \\ 1_L \\ 2_H \\ 2_L \\ \vdots \\ n_H \\ n_L \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & \cdots & n \end{array} \\ \begin{pmatrix} \mu_1 & 0 & 0 & \cdots & 0 \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} \mu_1 \\ \alpha_1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \cdots & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} \mu_1 \\ \alpha_1 \end{bmatrix} & \cdots & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \cdots & \begin{bmatrix} \mu_1 \\ \alpha_1 \end{bmatrix} \end{pmatrix} \end{array}.$$

As for the off-diagonal submatrices, we have:

$Q_{1,2} = Q_{2,3} = \cdots = Q_{m-1,m} = \lambda_1 I_{2n+1}$. Also, for $i = 2, 3, \ldots, m$:

$$Q_{i,i-1} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & \cdots & n-1 & n \end{array} \\ \begin{pmatrix} \beta_{i-1} & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} & \underline{\mathbf{0}} \\ \underline{\mathbf{0}}' & A_i & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \underline{\mathbf{0}}' & \mathbf{0} & A_i & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \underline{\mathbf{0}}' & \mathbf{0} & \mathbf{0} & \cdots & A_i & \mathbf{0} \\ \underline{\mathbf{0}}' & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A_i \end{pmatrix} \end{array},$$

with $\beta_i = \mu_1 + i\alpha_1$, $A_i = \begin{bmatrix} \beta_{i-1} & 0 \\ 0 & i\alpha_1 \end{bmatrix}$, $\mathbf{0} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ and $\underline{\mathbf{0}} = \begin{bmatrix} 0 & 0 \end{bmatrix}$.

Finally, for the diagonal submatrices, we obtain:

$$Q_{i,i} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & \cdots & n-1 & n \end{array} \\ \begin{pmatrix} -(\lambda+\beta_{i-1}) & \underline{e}\Lambda_2 & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} & \underline{\mathbf{0}} \\ B_1\underline{e}_1' & C_{i,1} & \Lambda_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \underline{\mathbf{0}}' & B_2 & C_{i,2} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \underline{\mathbf{0}}' & \mathbf{0} & \mathbf{0} & \cdots & C_{i,n-1} & \Lambda_2 \\ \underline{\mathbf{0}}' & \mathbf{0} & \mathbf{0} & \cdots & B_n & C_{i,n} \end{pmatrix} \end{array}, \; i = 1, 2, \ldots, m-1, \text{ and}$$

$$Q_{m,m} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & \cdots & n-1 & n \end{array} \\ \begin{pmatrix} -(\lambda_2+\beta_{m-1}) & \underline{e}\Lambda_2 & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} & \underline{\mathbf{0}} \\ B_1\underline{e}_1' & D_1 & \Lambda_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \underline{\mathbf{0}}' & B_2 & D_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \underline{\mathbf{0}}' & \mathbf{0} & \mathbf{0} & \cdots & D_{n-1} & \Lambda_2 \\ \underline{\mathbf{0}}' & \mathbf{0} & \mathbf{0} & \cdots & B_n & D_n \end{pmatrix} \end{array},$$

where $\Lambda_2 = \begin{bmatrix} \lambda_2 & 0 \\ 0 & \lambda_2 \end{bmatrix}$, $\underline{\mathbf{e}} = (1, 1)$, $\underline{\mathbf{e_1}} = (1, 0)$ and $B_i = \begin{bmatrix} i\alpha_2 & 0 \\ \mu_2 & (i-1)\alpha_2 \end{bmatrix}$

Also, we have for $j = 1, 2, \ldots, n-1$:

$$C_{i,j} = \begin{bmatrix} -(\lambda + j\alpha_2 + \beta_{i-1}) & 0 \\ 0 & -(\lambda + i\alpha_1 + \gamma_{j-1}) \end{bmatrix} \text{ and } D_j = \begin{bmatrix} -(\lambda_2 + j\alpha_2 + \beta_{m-1}) & 0 \\ 0 & -(\lambda_2 + m\alpha_1 + \gamma_{j-1}) \end{bmatrix}.$$

Finally, we have:

$$C_{1,n} = \begin{bmatrix} -(\lambda_1 + n\alpha_2 + \beta_{i-1}) & 0 \\ 0 & -(\lambda_1 + i\alpha_1 + \gamma_{n-1}) \end{bmatrix} \text{ and } D_n = \begin{bmatrix} -(n\alpha_2 + \beta_{m-1}) & 0 \\ 0 & -(m\alpha_1 + \gamma_{n-1}) \end{bmatrix}.$$

Note that Q is what is known as a level-dependent quasi-birth-and-death (QBD) process. This is reflected by having the sub-diagonal, main-diagonal and super-diagonal block matrices different for each row (level). The main cause of this is the inclusion of redistribution in the model. Should the model be free from reneging (i.e. $\alpha_1 = \alpha_2 = 0$), we would simply end up with a pure QBD process. And so, a useful observation of expressing Q in the form of these block matrices is noticing properties, as such, that are already recognized in the literature.

However, next to this, and the need to be able to write down the elements of this infinitesimal generator matrix on paper in a presentable form, the main motivation behind representing Q in the form of these block matrices was to avoid the necessity of having to solve $mn+(m+1)(n+1)$ linear equations simultaneously, which would lead to stability-related issues for large values of m and n. The next section shows how [1] uses the above mentioned submatrices in formulating a more computationally effective way to solve the equilibrium equations.

2.2 Solving the Equilibrium Equations

Using the previously mentioned block matrices and letting $\underline{\mathbf{0}}_{1,j}$ represent a 1-by-j vector of zeroes, one can break down $\underline{\pi}Q = \underline{\tilde{\mathbf{0}}}$ into the following equations:

1. $\underline{\mathbf{0}}_{1,n+1} = \underline{\pi}_0 Q_{0,0} + \underline{\pi}_1 Q_{1,0}$.

2. $\underline{\mathbf{0}}_{1,2n+1} = \underline{\pi}_0 Q_{0,1} + \underline{\pi}_1 Q_{1,1} + \underline{\pi}_2 Q_{2,1}$.

3. For $i = 1, 2, \ldots, m-1$:
   $\underline{\mathbf{0}}_{1,2n+1} = \lambda_1 \underline{\pi}_{i-1} + \underline{\pi}_i Q_{i,i} + \underline{\pi}_{i+1} Q_{i+1,i}$.

4. $\underline{\mathbf{0}}_{1,2n+1} = \lambda_1 \underline{\pi}_{m-1} + \underline{\pi}_m Q_{m,m}$.

Using an inductive approach, one can verify the following recursive expression derived in [1]:

$$\underline{\pi}_i = (-1)^i \underline{\pi}_0 \prod_{k=1}^{i} S_k,\ i = 1, 2, ..., m,$$

$$\text{where } S_j = \begin{cases} Q_{0,1}(Q_{1,1} - S_2 Q_{2,1})^{-1} & , j = 1 \\ \lambda_1 (Q_{j,j} - S_{j+1} Q_{j+1,j})^{-1} & , j = 2, 3, ..., m-1 \\ \lambda_1 Q_{m,m}^{-1} & , j = m \end{cases}$$

By setting $S_0 = Q_{0,0} - S_1 Q_{1,0}$ and knowing that $\sum_{i=0}^{m} \sum_{j=0}^{n} \pi_{i,j} = 1$, one can calculate $\underline{\pi}_0$, and hence calculate all the steady-state probabilities.

Therefore, using these derivations, the computational complexity is reduced significantly from dealing with $(mn + (m+1)(n+1))$ simultaneous equations to recursive arguments with matrices of a maximum dimension of $(2n+1)$-by-$(2n+1)$. Following this model, insurance companies can use these probabilities in assessing the efficiency and effectiveness of their claim processing system. For instance, by knowing that if the priority-based queues for a certain employee is usually empty because she/he assesses the claim efficiently, the insurance company can decide to reassign claims that are of a more time consuming nature. Another example could be if the queues are full most of the time (i.e. π_{m,X_2} and/or $\pi_{X_1,n}$ are/is high), the insurance company can judge whether the employee is slow in processing these claims relative to the entrance rate of new ones into the system. It might be decided that the employee be replaced or retained, or that some of these claims can be redistributed.

Next, we proceed to deriving the waiting-time distributions of HP claims.

Chapter 3

Waiting-time Distributions for HP Claims

3.1 Methodology

The objective here is to derive $F_{T^H|NotF}(\omega)$, $E[T^H|NotF]$ and $Var(T^H|NotF)$. This is done through the following steps (for a description of the following expressions, please refer to Table 1.1):

1. Derive $F_{W^H_{X_1,X_2}}(\omega)$.
2. Derive $F_{W^H}(\omega)$.
3. Derive $F_{T^H}(\omega)$.
4. Derive $F_{T^H|NotF}(\omega)$, $E[T^H|NotF]$ and $Var(T^H|NotF)$.

The reason why we are interested in the above conditional random variable is that we want to exclude the possibility that the HP queue is full upon arrival of our arbitrary HP claim, which also (technically speaking) yields a waiting-time equal to zero (the other possibility is having both queues empty, implying immediate entry into service).

To avoid repetition in the following sections, we assume that ω only takes on non-negative values.

3.2 Deriving $F_{W^H_{X_1,X_2}}(\omega)$

By conditioning on the distribution of certain values of X_1, X_2 and P, we obtain the following four exhaustive scenarios for $W^H_{X_1,X_2}$:

1. Case 1: $(X_1 = 0, X_2 = 0)$ or $(X_1 = m, X_2 \geq 0)$.
2. Case 2: $(X_1 = 0, X_2 \geq 1, P = L)$.
3. Case 3: $(1 \leq X_1 \leq m-1, X_2 \geq 0, P = H)$.
4. Case 4: $(1 \leq X_1 \leq m-1, X_2 \geq 1, P = L)$.

We derive the cdf for each case.

3.2.1 Case 1: $(X_1 = 0, X_2 = 0)$ or $(X_1 = m, X_2 \geq 0)$

In this case, either the HP claim will be immediately processed upon entry into the system, or the HP claim will be blocked from the system due to the HP queue being full. Therefore, in both situations, the waiting-times will be zero (i.e. $F_{W^H_{0,0}}(\omega) = F_{W^H_{m,X_2}}(\omega) = 1$).

3.2.2 Case 2: $(X_1 = 0, X_2 \geq 1, P = L)$

Since there are no claims in the HP queue, the newly arriving HP claim will get processed immediately after the employee finishes processing the current LP claim.

Thus, $W^H_{0,X_2} \sim Exp(\mu_2) \Leftrightarrow F_{W^H_{0,X_2,L}}(\omega) = 1 - e^{-\mu_2 \omega}$.

Note that the number of remaining claims in the LP queue after the one being processed leaves the system is irrelevant in the calculation of the distribution.

3.2.3 Case 3: $(1 \leq X_1 \leq m-1, X_2 \geq 0, P = H)$

In this case, $W^H_{X_1,X_2,H}$ can be expressed as a sum of independent exponential random variables: $\sum_{i=1}^{X_1} Y_i^H$.

Each Y_i^H can be interpreted as the first "impactful" event to occur with respect to the ith claim in the HP queue, which is either the redistribution of one of the waiting claims in front of and including it or the service completion of the HP claim being processed. In essence, $W^H_{X_1,X_2,H}$ is the total time required to remove the X_1 claims ahead of the arriving HP claim.

As a result, one can show that $W^H_{X_1,X_2,H}$ follows a hypoexponential distribution (for a detailed definition of hypoexponential distributions, we refer the reader to 7.6.3 of [5], pages 162-164). This is done by first showing that the Y_i^H's are indeed exponentially distributed and independent. Since Y_i^H is equal to the minimum of $(i-1)$ independent exponentially distributed redistribution times and an independent exponentially distributed service time, $Y_i^H \sim Exp(\mu_1 + (i-1)\alpha_1)$.

As a member of the phase-type family of distributions, referring to [2] and [4], one can write $W^H_{X_1,X_2,H} \sim PH_{X_1}(\underline{\alpha}, T_{X_1})$, so that:

$F^H_{W_{X_1,X_2,H}}(\omega) = Pr\{W^H_{X_1,X_2,H} \leq \omega\} = 1 - \underline{\alpha} exp\{T_{X_1}\omega\}\underline{e}'$, with

$$T_{X_1} = \begin{pmatrix} -(\mu_1 + (X_1-1)\alpha_1) & (\mu_1 + (X_1-1)\alpha_1) & 0 & \cdots & 0 \\ 0 & -(\mu_1 + (X_1-2)\alpha_1) & (\mu_1 + (X_1-2)\alpha_1) & \cdots & 0 \\ 0 & 0 & -(\mu_1 + (X_1-3)\alpha_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\mu_1 \end{pmatrix}.$$

We note that T_{X_1} is an X_1-by-X_1 matrix, $\underline{\alpha}$ is a 1-by-X_1 vector equal to $(1, 0, 0, \ldots, 0)$ and $\underline{e}'$ is an X_1-by-1 vector of ones.

3.2.4 Case 4: $(1 \leq X_1 \leq m-1, X_2 \geq 1, P = L)$

First of all, we define the following quantities:

1. $p_1(i)$**:** The probability that service to an LP claim (whom the employee is currently processing) is faster than the redistribution of one of i HP claims in queue. Hence, it follows that $p_1(i) = \frac{\mu_2}{\mu_2 + i\alpha_1}$.

2. Y_i^L**:** Analogous to Y_i^H in Case 3, we interpret Y_i^L as the first "impactful" event to occur with respect to the ith claim in the HP queue, given that an LP claim is currently in service. This event can either be the redistribution of one of the waiting HP claims in front of and including the ith HP claim, or the service completion of the LP claim being processed.

   Since the HP redistribution and LP service rates are exponentially distributed in this model, one can show that $Y_i^L \sim Exp(i\alpha_1 + \mu_2)$.

Using the definitions above, this case has the following initial mixture representation, based on two possible scenarios:

$$W^H_{X_1,X_2,L} = \begin{cases} Y^L_{X_1} + W^H_{X_1,X_2-1,H} & \text{with probability } p_1(X_1) \\ \\ Y^L_{X_1} + W^H_{X_1-1,X_2,L} & \text{with probability } 1 - p_1(X_1) \end{cases}$$

By dividing the second scenario into all possible sub-scenarios, one can derive the following representation:

$$W^H_{X_1,X_2,L} = \begin{cases} Y^L_{X_1} + W^H_{X_1,X_2-1,H} & \text{with probability } p_1(X_1) \\ Y^L_{X_1} + Y^L_{X_1-1} + W^H_{X_1-1,X_2-1,H} & \text{with probability } p_1(X_1-1)[1-p_1(X_1)] \\ \vdots & \\ \{\sum_{i=0}^{k} Y^L_{X_1-i}\} + W^H_{X_1-k,X_2-1,H} & \text{with probability } p_1(X_1-k)\prod_{i=0}^{k-1}[1-p_1(X_1-i)] \\ & k \in \{2,3,\ldots,X_1-1\} \\ \vdots & \\ \{\sum_{i=0}^{X_1} Y^L_{X_1-i}\} & \text{with probability } \prod_{i=1}^{X_1}[1-p_1(i)] \end{cases}$$

In distribution, we have:

$$W^H_{X_1,X_2,L} \sim \begin{cases} Exp(X_1\alpha_1+\mu_2) + PH_{X_1}(\underline{\alpha}, T_{X_1}) = PH_1(\underline{\beta}, \widetilde{T}_{X_1}) + PH_{X_1}(\underline{\alpha}, T_{X_1}) \\ Exp(X_1\alpha_1+\mu_2) + Exp((X_1-1)\alpha_1+\mu_2) + PH_{X_1-1}(\underline{\alpha}, T_{X_1-1}) = PH_2(\underline{\beta}, \tilde{T}_{X_1}) + PH_{X_1-1}(\underline{\alpha}, T_{X_1-1}) \\ \vdots \\ \sum_{i=0}^{X_1} Exp((X_1-i)\alpha_1+\mu_2) = PH_{X_1+1}(\underline{\beta}, \widetilde{T}_{X_1}) \end{cases}$$

Similar to Case 3, for a $PH_k(\underline{\alpha}, T_k)$ distribution, we have:

$$T_k = \begin{pmatrix} -(\mu_1 + (k-1)\alpha_1) & (\mu_1 + (k-1)\alpha_1) & 0 & \cdots & 0 \\ 0 & -(\mu_1 + (k-2)\alpha_1) & (\mu_1 + (k-2)\alpha_1) & \cdots & 0 \\ 0 & 0 & -(\mu_1 + (k-3)\alpha_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -\mu_1 \end{pmatrix}.$$

We note that T_k is a k-by-k matrix, $\underline{\alpha}$ is a 1-by-k vector equal to $(1, 0, 0, \ldots, 0)$ and $\alpha_0 = 0$.

As for a $PH_k(\underline{\beta}, \widetilde{T}_{X_1})$ distribution, we have:

$$\widetilde{T}_{X_1} = \begin{pmatrix} -(\mu_2 + X_1\alpha_1) & (\mu_2 + X_1\alpha_1) & 0 & \cdots & 0 \\ 0 & -(\mu_2 + (X_1-1)\alpha_1) & (\mu_2 + (X_1-1)\alpha_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -(\mu_2 + (X_1-k+1)\alpha_1) \end{pmatrix}.$$

In this case, we note that $\widetilde{T}_{X_1}$ is also a k-by-k matrix and $\underline{\beta} = \underline{\alpha}$.

From these results, one can deduce that:

$F^H_{W_{X_1,X_2,L}} = 1 - \underline{\eta} exp\{D_{2X_1+1}\omega\}\underline{e}'$, where

$\underline{e}' = (1, 1, \ldots, 1)'$ is a $(2X_1 + 1)$-by-1 vector, and $\underline{\eta}$ is a 1-by-$(2X_1 + 1)$ vector equal to $(1, 0, 0, \ldots, 0)$.

D_{2X_1+1} is a $(2X_1 + 1)$-by-$(2X_1 + 1)$ matrix, which can be partitioned into the following submatrices:

$$\begin{pmatrix} \widetilde{D}_{1,1} & \widetilde{D}_{1,2} \\ \underline{\mathbf{0}} & T_{X_1} \end{pmatrix}.$$

Here, $\underline{\mathbf{0}}$ is an X_1-by-$(X_1 + 1)$ matrix of zeroes, $\widetilde{D}_{1,1}$ is an $(X_1 + 1)$-by-$(X_1 + 1)$ matrix equal to

$$\begin{pmatrix} -(\mu_2 + X_1\alpha_1) & (1 - p_1(X_1))(\mu_2 + X_1\alpha_1) & 0 & \cdots & 0 \\ 0 & -(\mu_2 + (X_1 - 1)\alpha_1) & (1 - p_1(X_1 - 1))(\mu_2 + (X_1 - 1)\alpha_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -\mu_2 \end{pmatrix},$$

and $\widetilde{D}_{1,2}$ is an $(X_1 + 1)$-by-X_1 matrix equal to

$$\begin{pmatrix} p_1(X_1)(\mu_2 + X_1\alpha_1) & 0 & 0 & \cdots & 0 & 0 \\ 0 & p_1(X_1 - 1)(\mu_2 + (X_1 - 1)\alpha_1) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & p_1(1)(\mu_2 + \alpha_1) \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

Finally, note that in the special case when the exponential service rates of both claim types are equal (i.e. $\mu_1 = \mu_2 = \mu$), we can express $W^H_{X_1,X_2,L}$ as a sum of independent exponential random variables $\sum_{i=0}^{X_1} V_i$, where $V_i \sim Exp(i\alpha_1 + \mu)$. Thus, we will be dealing with a purely Case 3 situation, with $X_1 + 1$ claims instead of X_1 (i.e. $F^H_{W_{X_1,X_2,L}}(\omega) = F^H_{W_{X_1+1,X_2,H}}(\omega)$). In other words, having the service rates equal and the HP queue is not empty, one can consider the LP claim currently in service as being a member of the HP queue.

3.2.5 Rewriting the Cases in Phase-type Form

Because phase-type distributions in general do not have unique representations, the cdfs derived in the last three cases can be rewritten in such a way that the generator matrices and associated initial probability vectors are of equal dimension, for all values of X_1 and X_2 (i.e. Case i has a phase-type representation $PH_k(\underline{\alpha}_i, D_i)$, where $i = 2, 3, 4$, for some common dimension k). The main motivation behind this is that these cases are components of the unconditional waiting-time random variable W^H. By equalizing the dimensionality of the generator matrices and corresponding initial probability vectors, deriving $F_{W^H}(\omega)$ becomes simpler, which is the subject of the next section.

The cdfs of the last three cases can be rewritten as follows:

1. **Case 2:** $F_{W_{0,X_2}}(\omega) = 1 - \underline{\eta}_2 exp\{D_{2m-1}\omega\}\underline{e}'$,
   with $\underline{\eta}_2 = (0, 0, \ldots, 1, 0, \ldots, 0)$ at the mth position $\big($i.e. the last element of the $\widetilde{D}_{1,1}$ matrix: $-\mu_2\big)$, implying that the cdf can be re-collapsed into $1 - e^{-\mu_2\omega}$.

2. **Case 3:** $F^H_{W_{X_1,X_2,H}}(\omega) = 1 - \underline{\eta}_{3,X_1} exp\{D_{2m-1}\omega\}\underline{e}'$,
   with $\underline{\eta}_{3,X_1} = (0, 0, \ldots, 1, 0, \ldots, 0)$ at the $(2m - X_1)$th position $\big($i.e. the $(m - X_1)$th diagonal element of the T_{m-1} matrix: $-(\mu_1 + (X_1 - 1)\alpha_1)\big)$, implying that the cdf can be re-collapsed into $1 - \underline{\alpha} exp\{T_{X_1}\omega\}\underline{e}'$.

3. **Case 4:** $F^H_{W_{X_1,X_2,L}}(\omega) = 1 - \underline{\eta}_{4,X_1} exp\{D_{2m-1}\omega\}\underline{e}'$,
   with $\underline{\eta}_{4,X_1} = (0, 0, \ldots, 1, 0, \ldots, 0)$ at the $(m - X_1)$th position $\big($i.e. the $(m - X_1)$th diagonal element of the $\widetilde{D}_{1,1}$ matrix: $-(\mu_2 + X_1\alpha_1)\big)$, implying that the cdf can be re-collapsed into $1 - \underline{\eta} exp\{D_{2X_1+1}\omega\}\underline{e}'$.

Note that the rate transition matrices use the largest allowable value of X_1, which is $m - 1$, so that the claim can enter the system. Also, observe that the rate transition matrices in these cases are identical, which further facilitates the derivation of the (unconditional) waiting-time distribution of an arbitrarily arriving HP claim, as shown in the next section.

3.3 Deriving $F_{W^H}(\omega)$

Using the Law of Total Probability, we have:

$$F_{W^H}(\omega) = Pr\{W^H \leq \omega\}$$

$$= \sum_{i=0}^{m}\sum_{j=0}^{n} Pr\{W^H \leq \omega | X_1 = i, X_2 = j\}\pi_{i,j}$$

$$= \sum_{i=0}^{m}\sum_{j=0}^{n} Pr\{W^H \leq \omega | X_1 = i, X_2 = j\}\pi_{i,j}$$

$$= \pi_{0,0} + \sum_{j=0}^{n}\pi_{m,j} + \sum_{j=1}^{n}\pi_{0,j}Pr\{W_{0,j} \leq \omega\} + \sum_{i=1}^{m-1}\pi_{i,0}Pr\{W^H_{i,0,H} \leq \omega\}$$

$$+ \sum_{i=1}^{m-1}\sum_{j=1}^{n}\pi_{i,j,H}Pr\{W^H_{i,j,H} \leq \omega\} + \sum_{i=1}^{m-1}\sum_{j=1}^{n}\pi_{i,j,L}Pr\{W^H_{i,j,L} \leq \omega\}.$$

Thus, using the phase-type representations derived in the previous section, we have in distribution:

$$W^H \sim (\pi_{0,0} + \sum_{j=0}^{n}\pi_{m,j}) \cdot 0 + \sum_{j=1}^{n}\pi_{0,j}PH_{2m-1}(\underline{\eta}_2, D_{2m-1}) + \sum_{i=1}^{m-1}\pi_{i,0}PH_{2m-1}(\underline{\eta}_{3,i}, D_{2m-1})$$

$$+ \sum_{i=1}^{m-1}\sum_{j=1}^{n}\pi_{i,j,H}PH_{2m-1}(\underline{\eta}_{3,i}, D_{2m-1}) + \sum_{i=1}^{m-1}\sum_{j=1}^{n}\pi_{i,j,L}PH_{2m-1}(\underline{\eta}_{4,i}, D_{2m-1}).$$

By having a mass-point at zero (Case 1) and a mixture of phase-type distributions (Cases 2, 3 and 4), all with a common generator matrix D_{2m-1}, we end up with another phase-type distribution with the same generator matrix and a modified initial probability vector, which takes into account all possible values of X_1, X_2 and P. In particular, we have:

$$F_{W^H}(\omega) = 1 - \underline{\kappa}_H exp\{D_{2m-1}\omega\}\underline{e}',$$

where:

$$\underline{\kappa}_H = \Big(\sum_{j=1}^{n} \pi_{m-1,j,L}, \sum_{j=1}^{n} \pi_{m-2,j,L}, \ldots, \sum_{j=1}^{n} \pi_{1,j,L}, \sum_{j=1}^{n} \pi_{0,j}, (\pi_{m-1,0} + \sum_{j=1}^{n} \pi_{m-1,j,H}), (\pi_{m-2,0} + \sum_{j=1}^{n} \pi_{m-2,j,H}), \ldots, (\pi_{1,0} + \sum_{j=1}^{n} \pi_{1,j,H}) \Big)$$

We note that $\underline{\kappa}_H$ is a 1-by-$(2m-1)$ vector, whose elements sum to $1 - \pi_{0,0} - \sum_{j=0}^{n} \pi_{m,j}$.

Because we still consider the waiting-time equal to zero, even when the HP queue is full, we need to condition on the queue not being full upon the arrival of our arbitrary HP claim. Before normalizing our probabilities in this regard, we derive $F_{T^H}(\omega)$ (from which we can obtain $F_{T^H|NotF}(\omega)$).

3.4 Deriving $F_{T^H}(\omega)$

As mentioned in Chapter 1, $T^H = min\{W^H, R^H\}$. By assumption, W^H is independent of R^H. Thus, one can easily show that T^H also follows a phase-type distribution such that:

$$\begin{aligned} F_{T^H}(\omega) &= 1 - Pr\{T^H > \omega\} \\ &= 1 - Pr\{min\{W^H, R^H\} > \omega\} \\ &= 1 - Pr\{W^H > \omega\} Pr\{R^H > \omega\} \\ &= 1 - \underline{\kappa}_H exp\{Q_H \omega\} \underline{e}'. \end{aligned}$$

Here, we have $Q_H = D_{2m-1} - \alpha_1 I_{2m-1}$.

Finally, we proceed to deriving the conditional distribution of T^H, given the HP queue is not full upon arrival.

3.5 Deriving $F_{T^H|NotF}(\omega)$, $E[T^H|NotF]$ and $Var(T^H|NotF)$

By conditioning on the probability that the HP queue is not full, one gets:

$F_{T^H|NotF}(\omega) = 1 - \underline{\kappa}_{H|NotF} exp\{Q_H \omega\}\underline{e}'$, where $\underline{\kappa}_{H|NotF} = \frac{\underline{\kappa}_H}{1-\sum_{j=0}^{n} \pi_{m,j}}$.

Consequently, using phase-type theory, we have:

- $E[T^H|NotF] = -\underline{\kappa}_{H|NotF} Q_H^{-1} \underline{e}'$.
- $Var[T^H|NotF] = 2\underline{\kappa}_{H|NotF} Q_H^{-2} \underline{e}' - (E[T^H|NotF])^2$.

The next chapter derives the corresponding waiting-time distribution of an arbitrarily arriving LP claim.

Chapter 4

Waiting-time Distributions for LP Claims

4.1 Methodology

Analogous to the the previous chapter on HP waiting-times, the objective here is to derive $F_{T^L|NotF}(\omega)$, $E[T^L|NotF]$ and $Var(T^L|NotF)$. This is done through the following steps (for a description of the following expressions, please refer to Table 1.1):

1. Derive $F_{W^L}(\omega)$.
2. Derive $F_{T^L}(\omega)$.
3. Derive $F_{T^L|NotF}(\omega)$, $E[T^L|NotF]$ and $Var(T^L|NotF)$.

As mentioned in the previous chapter, we are interested in the above conditional random variable because we want to exclude the possibility that the LP queue is full upon arrival of our arbitrary LP claim, which also (technically speaking) yields a waiting-time equal to zero (the other possibility is having both queues empty, implying immediate entry into service).

To avoid repetition in the following sections, we again assume that ω only takes on non-negative values.

4.2 Deriving $F_{W^L}(\omega)$

Due to the independent exponential rates in our model, one finds that W^L follows a phase-type distribution.

Specifically, we have:

$F_{W^L}(\omega) = 1 - \underline{\kappa}_L exp\{D\omega\}\underline{e}'$, where:

- $\underline{\kappa}_L = (\tilde{\underline{\kappa}}_{n-1}, \underline{\kappa}_{n-1}, \underline{\kappa}_{n-2}, \ldots, \underline{\kappa}_1)$, such that:

  1. $\tilde{\underline{\kappa}}_{n-1} = (\pi_{1,n-1,H},\ \pi_{2,n-1,H}, \ldots,\ \pi_{m,n-1,H})$.
  2. $\underline{\kappa}_1 = (\pi_{0,1},\ \pi_{1,1,L},\ \pi_{2,1,L}, \ldots,\ \pi_{m,1,L},\ \pi_{1,0},\ \pi_{2,0}, \ldots,\ \pi_{m,0})$.
  3. $\underline{\kappa}_i = (\pi_{0,i},\ \pi_{1,i,L},\ \pi_{2,i,L}, \ldots,\ \pi_{m,i,L},\ \pi_{1,i-1,H},\ \pi_{2,i-1,H}, \ldots,\ \pi_{m,i-1,H})$, for $i = 2, 3, ..., n-1$.

- $\underline{e}'$ is an $(nm + (n-1)(m+1))$-by-1 vector of ones.

- D is the generator matrix, having a square dimension of $(nm + (n-1)(m+1))$. It can be expressed as follows:

$$D = \begin{pmatrix} T_{n-1} & [A,\ (n-1)\alpha_2 I_m] & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & A_{n-1} & B_{n-1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & A_{n-2} & B_{n-2} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & A_{n-3} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & A_2 & B_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A_1 \end{pmatrix}.$$

For the generator matrix D, the bold zeroes represent zero-filled matrices of appropriate dimension.

The diagonal block matrices, $A_1,\ A_2, \ldots, A_{n-1}$, are $(2m+1)$-by-$(2m+1)$ matrices, which can be represented by the following submatrices:

$\begin{pmatrix} U_i & B \\ \mathbf{0} & T_{i-1} \end{pmatrix}$, where T_i is an m-by-m matrix, such that $T_i = T_0 - i\alpha_2 I_m$, with T_0 equal to

$$\begin{pmatrix} -(\lambda_1+\mu_1) & \lambda_1 & 0 & \cdots & 0 & 0 \\ \mu_1+\alpha_1 & -(\lambda_1+\mu_1+\alpha_1) & \lambda_1 & \cdots & 0 & 0 \\ 0 & \mu_1+2\alpha_1 & -(\lambda_1+\mu_1+2\alpha_1) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -(\lambda_1+\mu_1+(m-2)\alpha_1) & \lambda_1 \\ 0 & 0 & 0 & \cdots & \mu_1+(m-1)\alpha_1 & -(\mu_1+(m-1)\alpha_1) \end{pmatrix},$$

U_i is an $(m+1)$-by-$(m+1)$ matrix, such that $U_i = U_1 - (i-1)\alpha_2 I_{m+1}$, with U_1 equal to

$$\begin{pmatrix} -(\lambda_1+\mu_2) & \lambda_1 & 0 & \cdots & 0 & 0 \\ \alpha_1 & -(\lambda_1+\mu_2+\alpha_1) & \lambda_1 & \cdots & 0 & 0 \\ 0 & 2\alpha_1 & -(\lambda_1+\mu_2+2\alpha_1) & \cdots & 0 & 0 \\ 0 & 0 & 3\alpha_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -(\lambda_1+\mu_2+(m-1)\alpha_1) & \lambda_1 \\ 0 & 0 & 0 & \cdots & m\alpha_1 & -(\mu_2+m\alpha_1) \end{pmatrix},$$

and B is an $(m+1)$-by-m matrix equal to $\begin{pmatrix} \mathbf{0} \\ \mu_2 I_m \end{pmatrix}$.

As for the off-diagonal blocks, $[A,\ (n-1)\alpha_2 I_m]$ describes the column concatenation of the matrices A and $(n-1)\alpha_2 I_m$, where A is an m-by-$(m+1)$ matrix with μ_1 in the upper left corner, zero everywhere else. $B_2,\ B_3, \ldots, B_{n-1}$ are $(2m+1)$-by-$(2m+1)$ block matrices, which can be represented by the following submatrices:

$\begin{pmatrix} C_{i-1} & \mathbf{0} \\ A & (i-1)\alpha_2 I_m \end{pmatrix}$,

where C_i is an $(m+1)$-by-$(m+1)$ matrix equal to $\begin{pmatrix} \mu_2+i\alpha_2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & i\alpha_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & i\alpha_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & i\alpha_2 & 0 \\ 0 & 0 & 0 & \cdots & 0 & i\alpha_2 \end{pmatrix}$.

4.3 Deriving $F_{T^L}(\omega)$

As mentioned in Chapter 1, $T^L = min\{W^L, R^L\}$. By assumption, W^L is independent of R^L. Thus, one can easily show that T^L also follows a phase-type distribution such that:

$$\begin{aligned} F_{T^L}(\omega) &= 1 - Pr\{T^L > \omega\} \\ &= 1 - Pr\{min\{W^L, R^L\} > \omega\} \\ &= 1 - Pr\{W^L > \omega\} Pr\{R^L > \omega\} \\ &= 1 - \underline{\kappa}_L exp\{Q_L \omega\} \underline{e}'. \end{aligned}$$

Here, $Q_L = D - \alpha_2 I_{nm+(n-1)(m+1)}$.

Notice that for large values of m and n, the dimensionality of Q_L becomes significantly larger than Q_H derived in the previous chapter (which is of dimension $(2m+1)$). Thus, computational complexity for calculating the matrix exponential and the inverse of Q_L becomes an obstacle in obtaining numerical quantities when n and m are large.

We have successfully derived a methodology to reduce the computational complexity of the inverse of Q_L through a recursive method shown in the next page.

1. Since $Q_L = D - \alpha_1 I_{nm+(n-1)(m+1)}$, let:

   (a) For $i = 1, 2, \ldots, n-1$: $\hat{A}_i = A_i - \alpha_2 I_{2m+1}$.

   (b) $\hat{T}_{n-1} = T_{n-1} - \alpha_2 I_m$.

2. Set $Q_1 = \hat{A}_1 \Rightarrow Q_1^{-1} = \hat{A}_1^{-1}$.

3. Set $Q_2 = \begin{pmatrix} \hat{A}_2 & B_2 \\ \mathbf{0} & \hat{A}_1 \end{pmatrix} = \begin{pmatrix} \hat{A}_2 & B_2 \\ \mathbf{0} & Q_1 \end{pmatrix} \Rightarrow Q_2^{-1} = \begin{pmatrix} \hat{A}_2^{-1} & -\hat{A}_2^{-1} B_2 Q_2^{-1} \\ \mathbf{0} & Q_1^{-1} \end{pmatrix}$.

4. Set $Q_3 = \begin{pmatrix} \hat{A}_3 & B_3 & \mathbf{0} \\ \mathbf{0} & \hat{A}_2 & B_2 \\ \mathbf{0} & \mathbf{0} & \hat{A}_1 \end{pmatrix} = \begin{pmatrix} \hat{A}_3 & [B_3, \mathbf{0}] \\ \mathbf{0} & Q_2 \end{pmatrix} \Rightarrow Q_3^{-1} = \begin{pmatrix} \hat{A}_3^{-1} & -\hat{A}_3^{-1}[B_3, \mathbf{0}] Q_2^{-1} \\ \mathbf{0} & Q_2^{-1} \end{pmatrix}$.

   Note that the $\mathbf{0}$'s are of appropriate dimension and can differ from one another.

5. One continues along the same pattern, until:

$$Q_{n-1} = \begin{pmatrix} \hat{A}_{n-1} & [B_{n-1}, \mathbf{0}] \\ \mathbf{0} & Q_{n-2} \end{pmatrix} \Rightarrow Q_{n-1}^{-1} = \begin{pmatrix} \hat{A}_{n-1}^{-1} & -\hat{A}_{n-1}^{-1}[B_{n-1}, \mathbf{0}] Q_{n-2}^{-1} \\ \mathbf{0} & Q_{n-2}^{-1} \end{pmatrix}.$$

6. Finally, one can write Q_L as:

$$\begin{pmatrix} \hat{T}_{n-1} & [A, (n-1)\alpha_2 I, \mathbf{0}] \\ \mathbf{0} & Q_{n-1} \end{pmatrix}, \text{ and thus } Q_L^{-1} = \begin{pmatrix} \hat{T}_{n-1}^{-1} & -\hat{T}_{n-1}^{-1}[A, (n-1)\alpha_2 I, \mathbf{0}] Q_{n-1}^{-1} \\ \mathbf{0} & Q_{n-1}^{-1} \end{pmatrix}.$$

We remark that this technique can also be used in reducing the computational complexity of the rate transition matrix Q^H, derived in the previous chapter. We now proceed to deriving the conditional distribution of T^L, given the LP queue is not full upon arrival.

4.4 Deriving $F_{T^L|NotF}(\omega)$, $E[T^L|NotF]$ and $Var(T^L|NotF)$

By conditioning on the probability that the LP queue is not full, one gets:

$F_{T^L|NotF}(\omega) = 1 - \underline{\kappa}_{L|NotF} exp\{Q_L\omega\}\underline{e}'$, where $\underline{\kappa}_{L|NotF} = \frac{\underline{\kappa}_L}{1-\sum_{i=0}^{m}\pi_{i,n}}$.

Consequently, using phase-type theory, we have:

- $E[T^L|NotF] = -\underline{\kappa}_{L|NotF} Q_L^{-1}\underline{e}'$.
- $Var[T^L|NotF] = 2\underline{\kappa}_{L|NotF} Q_L^{-2}\underline{e}' - (E[T^L|NotF])^2$.

We now proceed to a simple numerical example to give an idea on how such as system could be beneficial.

Chapter 5

Illustration

5.1 Simple Numerical Example

Assume an employee is assigned to the following system for the day, with specified rates and obtained conditional distributions shown in the following two diagrams:

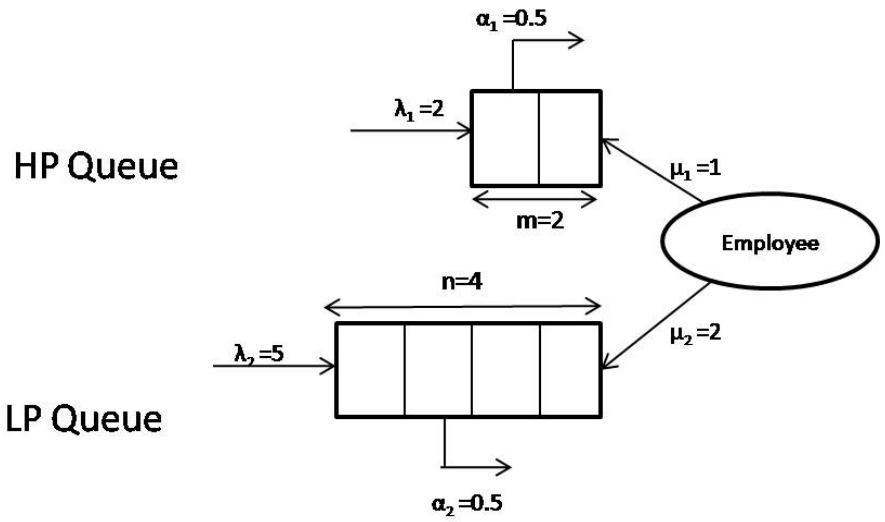

Figure 5.1: Numerical Example Parameters

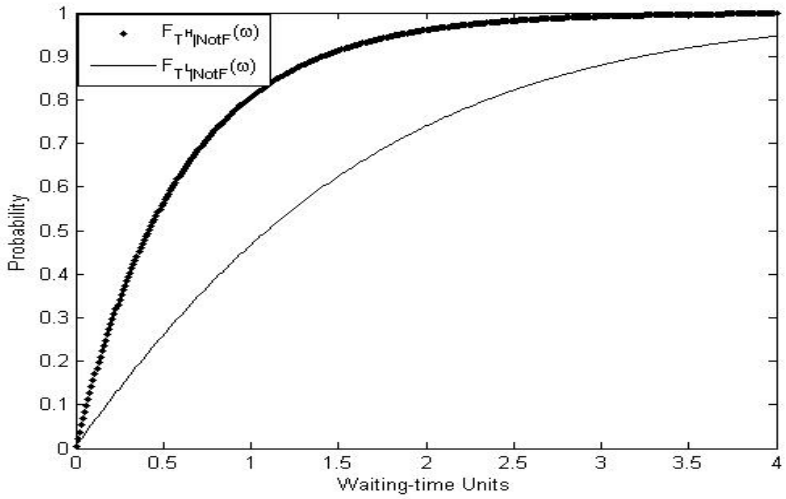

Figure 5.2: Comparing Conditional Times Spent in HP and LP Queues

All calculations here have been obtained using MATLAB. For details regarding the computer code used, please see **Appendix A**.

$\pi_{\mathbf{i,j}}$	$\mathbf{j=0}$	$\mathbf{j=1}$	$\mathbf{j=2}$	$\mathbf{j=3}$	$\mathbf{j=4}$
$\mathbf{i=0}$	0.0029	0.0091	0.0231	0.0466	0.0751
$\mathbf{i=1}$	0.0022	0.0111	0.0378	0.0985	0.2041
$\mathbf{i=2}$	0.0018	0.0119	0.0466	0.1329	0.2963

Table 5.1: Steady State Probabilities of Numerical Example

- $Pr\{HP\ Queue\ Not\ Full\} = 1 - \sum_{j=0}^{4} \pi_{2,j} \approx 0.5105$
- $Pr\{LP\ Queue\ Not\ Full\} = 1 - \sum_{i=0}^{2} \pi_{i,4} \approx 0.4246$
- $E[X_1] \approx 1.3327$.
- $E[X_2] \approx 3.3828$.
- $E[T^H|NotF] \approx 0.6104$.
- $E[T^L|NotF] \approx 1.4580$.
- $\sqrt{Var(T^H|NotF)} \approx 0.6245$
- $\sqrt{Var(T^L|NotF)} \approx 1.3282$

5.2 Possible Improvements on the Numerical Example

As the above calculations show, both queues have a high probability of being full, implying either that the employee is not assessing/processing the claims fast enough or the queueing buffers need to be increased. For instance, by having all the rates the same (i.e. $\lambda_1 = 2,\ \lambda_2 = 5,\ \mu_1 = 1,\ \mu_2 = 2,\ \alpha_1 = \alpha_2 = 0.5$) and doubling the limit size of both queues (i.e. $m = 4$ and $n = 8$), the chances of having a full queue are reduced:

- $Pr\{HP\ Queue\ Not\ Full\} = 1 - \sum_{j=0}^{8} \pi_{4,j} \approx 0.7910.$
- $Pr\{LP\ Queue\ Not\ Full\} = 1 - \sum_{i=0}^{4} \pi_{i,8} \approx 0.6922.$

In addition to this, if the employee, for instance, were replaced by a more proficient one who can process the claims at double the speed (i.e. $\mu_1 = 2$ and $\mu_2 = 4$), we witness a further improvement in performance, namely:

- $Pr\{HP\ Queue\ Not\ Full\} = 1 - \sum_{j=0}^{8} \pi_{4,j} \approx 0.9068.$
- $Pr\{LP\ Queue\ Not\ Full\} = 1 - \sum_{i=0}^{4} \pi_{i,8} \approx 0.7895.$
- $E[T^H|NotF] \approx 0.5187.$
- $E[T^L|NotF] \approx 1.3502.$

As a result, given the available resources, insurance companies following this setup will be better equipped in creating optimal claim processing systems.

Chapter 6

Concluding Remarks and Further Research

The derivations of this model are complete from the theoretical point of view. However, as pointed out in **Chapter 4**, we must consider the computational complexity of our expressions. Our final unfinished task in this model is deriving a method that significantly reduces the complexity of the matrix exponential of the generator matrix in the LP waiting-time case, especially for large values of m and n. Therefore, in order to maximize the potential of applying this model in practice (where buffers could hold over a hundred claims each), one has to find optimal methods for computing this matrix exponential.

Next to this, we also plan to derive all the previous distributions under a more general framework: we will work with the redistribution rates as functions of the claims' positions in the queue. We intend to begin by assessing the case where the redistribution rates are linearly decreasing functions of the position of the claim in the queue. In other words, the closer the claim is to being served, the smaller its redistribution rate becomes. One can find this scenario practically plausible if one argues that other departments in the insurance company become more reluctant to incorporate claims into their own system, the closer these claims are to be being processed in the original system.

To conclude, we find that using mathematical theory is a necessary tool for improving the effectiveness and efficiency of institutions and organizations. The more literatures broaden, the greater the chances that overlaps and synergies would occur between them. With improved models and increasing technological facilities, one aims to help guide us to new paradigms and horizons.

Appendix: MATLAB Code

```
m=2; n=4;lambda1=2;lambda2=5;mu1=1;mu2=2;alpha1=0.5;alpha2=0.5;%Initial Conditions

%1) Calculating Transition Probabilities

lambda=lambda1+lambda2;
%A) Gamma Matrix; Q(0,1)
Q=zeros(n+1,2*n+1);
for i=1:n+1
    for j=1:2*n+1
            if i==1 && j==1         %If we are in the first row
                    Q(i,j)=lambda1;
            else            %if we are not in the first row
                if j==2*i-1
                    Q(i,j)=lambda1;
                end
            end
    end
end
Gamma=Q;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%B)Psi_k Matrices; k=1,1,2,...,m; Q(k,k-1)
%i)Psi_1 Matrix; Q(1,0)
Q=zeros(2*n+1,n+1);
for i=1:2*n+1
    for j=1:n+1
        if i==1 && j==1
           Q(i,j)=mu1;
        else
            if i==2*(j-1)
                Q(i,j)=mu1;
                Q(i+1,j)=alpha1;
            end
        end
    end
end
Psi_1=Q;
%ii)Psi_k Matrices; k=2,3,...,m; Q(k,k-1)
if m>1
for k=2:m
 Q=zeros(2*n+1,2*n+1);
    for i=1:2*n+1
        for j=1:2*n+1
            if i==1 && j==1
                Q(i,j)=mu1+(k-1)*alpha1;
            else
                if i==j
```

```
                if i/2 == floor(i/2)%To distinguish between high and low priority
                    Q(i,j)=mu1+(k-1)*alpha1;
                else
                    Q(i,j)=k*alpha1;
                end
            end
        end
    end
end
if k==2
    Psi_k=Q;
else
    Psi_k=horzcat(Psi_k,Q);%Concatenating the Psi_k's in ascending order
end
end
else
    Psi_k=Psi_1;
end%End of if statement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%C)P_k Matrices; k=0,1,2,...,m;  Q(k,k)
%i)P_0 Matrix; Q(0,0)
Q=zeros(n+1,n+1);
for i=1:n+1
    for j=1:n+1
        if i==j
            if i==1
            Q(i,j)=-lambda;%First edge of the matrix
            elseif i==n+1
            Q(i,j)=-(lambda1+mu2+(i-2)*alpha2);%Last edge of the matrix
            else
            Q(i,j)=-(lambda+mu2+(i-2)*alpha2);
            end
        elseif i==j+1
                Q(i,j)=mu2+(j-1)*alpha2;
        elseif i==j-1
                Q(i,j)=lambda2;
        end
    end
end
P_0=Q;
%ii)P_k Matrices; k=1,2,...,m; Q(k,k)
for k=1:m
    Q=zeros(2*n+1,2*n+1);
    for i=1:2*n+1
        for j=1:2*n+1
            if i==j
                if i==1
                    if k==m
                    Q(i,j)=-(lambda2+mu1+(k-1)*alpha1);
                    else
                    Q(i,j)=-(lambda+mu1+(k-1)*alpha1);
                    end
                elseif i==2*n
                    if k==m
                    Q(i,j)=-(n*alpha2+mu1+(k-1)*alpha1);
                    else
                    Q(i,j)=-(lambda1+n*alpha2+mu1+(k-1)*alpha1);
                    end
```

```
                elseif i==2*n+1
                    if k==m
                    Q(i,j)=-(k*alpha1+mu2+(n-1)*alpha2);
                    else
                    Q(i,j)=-(lambda1+k*alpha1+mu2+(n-1)*alpha2);
                    end
                elseif i/2==floor(i/2)%make sure it is an even number
                    if k==m
                    Q(i,j)=-(lambda2+(j/2)*alpha2+mu1+(k-1)*alpha1);
                    else
                        if k==m
                        Q(i,j)=-(lambda2+(j/2)*alpha2+mu1+(k-1)*alpha1);
                        else
                        Q(i,j)=-(lambda+(j/2)*alpha2+mu1+(k-1)*alpha1);
                        end
                    end
                else
                    if k==m
                    Q(i,j)=-(lambda2+k*alpha1+mu2+((j-1)/2-1)*alpha2);
                    else
                    Q(i,j)=-(lambda+k*alpha1+mu2+((j-1)/2-1)*alpha2);
                    end
                end
            elseif i==1 && j==2
                Q(i,j)=lambda2;
            elseif j==i+2 && i/2==floor(i/2) %make sure it is an even number
                Q(i,j)=lambda2;
                Q(i+1,j+1)=lambda2;
            elseif i==2 && j==1
                Q(i,j)=alpha2;
                Q(i+1,j)=mu2;
            elseif i==j+2 && i/2==floor(i/2) %make sure it is an even number
                Q(i,j)=i/2*alpha2;
                Q(i+1,j)=mu2;
                Q(i+1,j+1)=(i/2-1)*alpha2;
            end
        end
    end
    if k==1
        P_k=Q;
    else
    P_k=horzcat(P_k,Q);%Concatenating the _k's in ascending order
    end
  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if m>1
%D)Calculating Sj; j=0,1,...,m
%i)Calculating Sj; j=1,...,m
    S=lambda1*inv(P_k(:,(2*n+1)*(m-1)+1:(2*n+1)*m));%value of Sm
    k=m-1;
    while(k>=2)%We are beginning from m-1 and working backwards
    %Taking into account that Psi_k starts from 2
    Q=lambda1*inv(P_k(:,(2*n+1)*(k-1)+1:(2*n+1)*k)-S(:,1:2*n+1)*Psi_k(:,(2*n+1)*(k-1)+1:(2*n+1)*(k)));

    S=horzcat(Q,S);%Concatenating the S's in ascending order starting from S_1
    k=k-1;
    end
    %Value of S1
```

```
    S_1=Gamma*inv(P_k(:,(2*n+1)*(k-1)+1:(2*n+1)*k)-S(:,(2*n+1)*(k-1)+1:(2*n+1)*k)*Psi_k(:,(2*n+1)*(k-1)+1:(2*n+1)*(k)));
%ii)Calculating S_0
    S_0=P_0-S_1*Psi_1;%Q takes the value of S1 from previous statement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%E)Calculating Pi_i's, i=0,1,2,...,m
%i)Calculating Pi_0
    n1=n+1; n2=2*n+1;
    u=ones(n1,1)-S_1*ones(n2,1);%Incorporating S1 from the start
    for i=2:m
    k=(-1)^i*S_1;%Putting the first two term of the product
    for j=2:i %As we're starting from S2, by already incorporating S1
        k=k*S(:,(2*n+1)*(j-2)+1:(2*n+1)*(j-1));%Recall that the S matrix starts from S2 not S1
    end
    u=u+k*ones(n2,1);
    end
    Coef_Pi_0=horzcat(S_0(:,1:n),u);
    RHS_Pi_0=horzcat(zeros(1,n),1);
    Pi_0=RHS_Pi_0*inv(Coef_Pi_0);
%ii)Calculating Pi_i's
    Pi_i=-Pi_0*S_1;%Calculating Pi_1
    for i=2:m
        k=(-1)^i*Pi_0*S_1;
        for j=2:i%As we're starting from S2, by already incorporating S1
            k=k*S(:,(2*n+1)*(j-2)+1:(2*n+1)*(j-1));
        end
        Pi_i=vertcat(Pi_i,k);
    end
else% if m==1
    P_0_Star=P_0; P_0_Star(:,n+1)=ones(n+1,1);%Making the last column ones
    Psi_1_Star=Psi_1; Psi_1_Star(:,n+1)=ones(2*n+1,1);%Making the last column ones
    RHS_Pi_0=horzcat(zeros(1,n),1);
    Pi_0=RHS_Pi_0*inv(P_0_Star-Gamma*inv(P_k)*Psi_1_Star);
    Pi_i=-Pi_0*Gamma*inv(P_k);
end%end of if statement, m>1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%F)Display of Pi_i's
Distr=zeros(m+1,n+1);
Distr(1,:)=Pi_0;
for i=2:m+1%We are starting from second row
    for j=1:n+1 %We are starting from first column
        if j==1
            Distr(i,j)=Pi_i(i-1,j);
        else
            Distr(i,j)=Pi_i(i-1,2*(j-1))+Pi_i(i-1,2*(j-1)+1);
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%G)Display of Expectations
EX1=0; EX2=0;
MarginX1=sum(Distr');%Marginal probabilities for X1
MarginX2=sum(Distr);%Marginal probabilities for X2
for i=1:m+1
    EX1=EX1+(i-1)*MarginX1(i);
end
for j=1:n+1
    EX2=EX2+(j-1)*MarginX2(j);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%2) Calculating HP Waiting-times

%A)Calculating F_{T^H}(omega)
%i) Calculating the C matrix, we assume that m>=2
D=zeros(2*m-1,2*m-1);
for i=1:2*m-1
    for j=1:2*m-1
        if i==j
            if i<=m%D1,1
               D(i,j)=-(mu2+(m-i)*alpha1);
            else%T
            D(i,j)=-(mu1+(2*m-i-1)*alpha1);
            end
        elseif j==i+1
             if i<=m%D1,1
               D(i,j)=(1-(mu2/(mu2+(m-i)*alpha1)))*(mu2+(m-i)*alpha1);
            else%T
            D(i,j)=-D(i,j-1);
             end
        elseif i<=m && j>=m+1 %D1,2
            if j==m+i
                D(i,j)=(mu2/(mu2+(m-i)*alpha1))*(mu2+(m-i)*alpha1);
            end
        end
    end
end
C=D-alpha1*eye(2*m-1);
%ii) Calculating the gamma matrix, we assume that m>=2
gamma=zeros(1,2*m-1);
for i=1:2*m-1
    for j=1:n
        if i<=m-1
        gamma(1,i)=gamma(1,i)+Pi_i(m-i,2*j+1);
        elseif i==m
        gamma(1,i)=gamma(1,i)+Pi_0(j+1);
        else
            if j==1
            gamma(1,i)=gamma(1,i)+Pi_i(2*m-i,1)+Pi_i(2*m-i,2*j);%to include the first term
            else
            gamma(1,i)=gamma(1,i)+Pi_i(2*m-i,2*j);
            end
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%B)Calculating E[T^H] and Var(T^H)
e=ones(2*m-1,1);
E_TH=-gamma*C^(-1)*e;
E_TH2=2*gamma*C^(-2)*e;
Var_TH=E_TH2-E_TH^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%C) Calculating F_{T^H}(omega)
scale=400;increments=1/100;
omega=zeros(1,scale+1);
omega(1)=0;
```

```
F_TH=zeros(1,scale+1);
F_TH(1)=1-gamma*expm(C*omega(1))*e;
for i=2:scale+1
omega(i)=(i-1)*increments;
F_TH(i)=1-gamma*expm(C*omega(i))*e;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%D) Conditioning on HP Queue Not Being Full
PrbH_NotF=1-sum(Distr(m+1,:));%Adding all columns of mth row (note that Distr starts at zero)
F_TH_NotF=1-(1-F_TH)/PrbH_NotF;

E_TH_NotF=E_TH/PrbH_NotF;
E_TH_NotF2=E_TH2/PrbH_NotF;
Var_TH_NotF=E_TH_NotF2-E_TH_NotF^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%E) Plotting Conditional Time Spent Probabilities for HP Claims
%plot(omega,F_TH_NotF)
%axis([0 max(omega) 0 1])
%xlabel('Time Units')
%ylabel('F_{T^H|InQ}(\omega)')
%title('Time Spent in HP Queue: m=2, n=4,\lambda_1=2,\lambda_2=5,\mu_1=1,\mu_2=2, \alpha_1=0.5, \alpha_2=0.5')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%3) Calculating LP Waiting-times

%A) Calculating Q
%m=2;n=4;lambda1=2;lambda2=5;mu1=1;mu2=2;alpha1=0.5;alpha2=0.5;
%i)Calculating T0 and U1
T0=zeros(m,m);%Initializing T0 matrix to zeros
U1=zeros(m+1,m+1);%Initializing U1 matrix to zeros
for i=1:m+1
    for j=1:m+1
        if (i<m+1&&j<m+1)%Gain of efficiency, since T0 is only an m by m matrix
            if j==i-1
            T0(i,i-1)=mu1+(i-1)*alpha1;
            U1(i,i-1)=(i-1)*alpha1;
            elseif j==i
                if i<m
                T0(i,i)=-(lambda1+(i-1)*alpha1+mu1);
                U1(i,i)=-(lambda1+(i-1)*alpha1+mu2);
                else
                T0(m,m)=-((m-1)*alpha1+mu1);%Last element,m, has no lambda1 term
                U1(m,m)=-(lambda1+(m-1)*alpha1+mu2);%No change for U1 term
                end
            elseif j==i+1
                T0(i,i+1)=lambda1;
                U1(i,i+1)=lambda1;
            end
        else
        U1(m,m+1)=lambda1;
        U1(m+1,m)=m*alpha1;
        U1(m+1,m+1)=-(m*alpha1+mu2);
        end
    end
end
A=zeros(m,m+1);
A(1,1)=mu1;
```

```
B=vertcat(zeros(1,m),eye(m));
B=B*mu2;

%ii)Calculating block matrices (Call Ai function and Bi function)

if n==1 Q=T0; QQ=Q-alpha2*eye(m);QQ_inv=inv(QQ);%QQ is to be the matrix: Q-Ialpha2
else
 An=T0-(n-1)*alpha2*eye(m);%Upper corner block matrix T_{n-1}
 An_QQ=An-alpha2*eye(m);
 Bn=[A (n-1)*alpha2*eye(m)];
 zeroH=zeros(2*m+1,2*m+1);%Zero matrices other than first row and first column
 zero_r1=zeros(m,2*m+1);%Zero matrices in first row
 zero_c1=zeros(2*m+1,m);%Zero matrices in first row
 if n==2
     temp1=Ai(U1,B,T0,1,m,alpha2);temp2=temp1-alpha2*eye(2*m+1);%Reduce notation
     Q=vertcat([An Bn],[zero_c1 temp1]);
     QQ=vertcat([An_QQ Bn],[zero_c1 temp2]);
     QQ_inv=vertcat([inv(An_QQ) -inv(An_QQ)*Bn*inv(temp2)],[zero_c1 inv(temp2)]);
 else
    tempA1=Ai(U1,B,T0,1,m,alpha2);tempAi=Ai(U1,B,T0,2,m,alpha2);tempBi=Bi(A,2,m,mu2,alpha2);
    tempA1QQ=tempA1-alpha2*eye(2*m+1);tempAiQQ=tempAi-alpha2*eye(2*m+1);
    Q=vertcat([tempAi tempBi],[zeroH tempA1]);%Starting point of recursion
    QQ=vertcat([tempAiQQ tempBi],[zeroH tempA1QQ]);%Starting point of recursion
    QQ_inv=vertcat([inv(tempAiQQ) -inv(tempAiQQ)*tempBi*inv(tempA1QQ)],[zeroH inv(tempA1QQ)]);
    zero_R1=zero_r1;%Concatenates first row zero matrices
    zero_C1=vertcat(zero_c1,zero_c1);%Concatenates first column zero matrices
    if n>3
        zeroUp=zeroH;
        zeroLeft=vertcat(zeroH,zeroH);
        for i=3:n-1
            tempAi=Ai(U1,B,T0,i,m,alpha2);tempBi=Bi(A,i,m,mu2,alpha2);tempAiQQ=tempAi-alpha2*eye(2*m+1);%Reduce notation
            Q=vertcat([tempAi tempBi zeroUp],[zeroLeft Q]);
            QQ=vertcat([tempAiQQ tempBi zeroUp],[zeroLeft QQ]);
            QQ_inv=vertcat([inv(tempAiQQ) -inv(tempAiQQ)*[tempBi zeroUp]*QQ_inv],[zeroLeft QQ_inv]);
            zeroUp=[zeroUp zeroH];
            zeroLeft=vertcat(zeroLeft,zeroH);
            zero_R1=[zero_R1 zero_r1];
            zero_C1=vertcat(zero_C1,zero_c1);
        end
    end%End of if n>3
    Q=vertcat([An Bn zero_R1],[zero_C1 Q]);
    QQ=vertcat([An_QQ  Bn zero_R1],[zero_C1 QQ]);
    QQ_inv=vertcat([inv(An_QQ) -inv(An_QQ)*[Bn zero_R1]*QQ_inv],[zero_C1 QQ_inv]);
 end%End of nested if-else statement
end%End of if-else statement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%B)Calculating Kappas
if n==1
    kappa=Pi_i(1,1);
    for i=2:m
        kappa=[kappa Pi_i(i,1)];%[pi_{1,0} pi_{2,0},...,pi_{m,0}]
    end
else
kappa_n=Pi_i(1,2*(n-1));
for i=2:m
kappa_n=[kappa_n Pi_i(i,2*(n-1))];%[pi_{1,n-1,H} pi_{2,n-1,H},...,pi_{m,n-1,H}]
end
```

```
kappa=kappa_n;
for i=2:n-1
kappa=[kappa kappai(Pi_0,Pi_i,(n-i+1),m)];%We have all the kappas, except for kappa1
end

%Calculating kappa1

kappa1=zeros(1,2*m);
for j=1:m
     kappa1(j)=Pi_i(j,3);%First m elements excluding pi_{0,i}. All pi_{j,i,L}
     kappa1(j+m)=Pi_i(j,1);%Last m elements excluding. All pi_{j,i,H}
end
kappa1=[Pi_0(2) kappa1];

%Concatenating kappa1 at the end

kappa=[kappa kappa1];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%C)Calculating E[T^L] and Var(T^L)
e=ones(m*n+(n-1)*(m+1),1);
E_TL=-kappa*QQ^(-1)*e;
E_TL2=2*kappa*QQ^(-2)*e;
Var_TL=E_TL2-E_TL^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%D) Calculating F_{T^L}(omega)
%scale=400;increments=1/100;
%omega=zeros(1,scale+1);
%omega(1)=0;
F_TL=zeros(1,scale+1);
F_TL(1)=1-kappa*expm(QQ*omega(1))*e;
for i=2:scale+1
omega(i)=(i-1)*increments;
F_TL(i)=1-kappa*expm(QQ*omega(i))*e;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%E) Conditioning on LP Queue Not Being Full
PrbL_NotF=1-sum(Distr(:,n+1));%Adding all columns of mth row (note that Distr starts at zero)
F_TL_NotF=1-(1-F_TL)/PrbL_NotF;

E_TL_NotF=E_TL/PrbL_NotF;
E_TL_NotF2=E_TL2/PrbL_NotF;
Var_TL_NotF=E_TL_NotF2-E_TL_NotF^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%vi)Plotting F_{T^L|NotFull}(omega)
%plot(omega,F_TL_NotF)
%axis([0 max(omega) 0 1])
%xlabel('\omega')
%ylabel('F_{T^L|InQ}(\omega)')
%title('Time Spent: m=2, n=4,\lambda_1=2,\lambda_2=5,\mu_1=1,\mu_2=2, \alpha_1=0.5, \alpha_2=0.5')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%4) Comparing F_TH|InQ and F_TL|InQ

plot(omega,F_TH_NotF,'k.',omega,F_TL_NotF,'k')
axis([0 max(omega) 0 1])
xlabel('Waiting-time Units')
```

```
ylabel('Probability')
legend('F_{T^H|NotF}(\omega)','F_{T^L|NotF}(\omega)')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%File Name: Ai.m
%Producing Ai's, i=1,2,...,n-1
function[X] = Ai(U1,B,T0,i,m,alpha2)
Ui=U1-(i-1)*alpha2*eye(m+1);
Ti=T0-(i-1)*alpha2*eye(m);
X=vertcat([Ui B],[zeros(m,m+1) Ti]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%File Name: Bi.m
%Producing Bi's, i=2,3,...,n-1
function[X] = Bi(A,i,m,mu2,alpha2)
Ci=eye(m+1)*(i-1)*alpha2;
Ci(1,1)=Ci(1,1)+mu2;
X=vertcat([Ci zeros(m+1,m)],[A eye(m)*(i-1)*alpha2]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%File Name: kappai.m
%Producing Low Priority Claim Block Vectors, Excluding First Block and Last block
%kappa=[kappa_n, kappa_{n-1},...,kappa_1]
%This Function Produces kappa_i, for i=2,...,n-1.
function[X] = kappai(Pi_0,Pi_i,i,m)
X=zeros(1,2*m);
for j=1:m
     X(j)=Pi_i(j,2*i+1);%First m elements excluding pi_{0,i}. All pi_{j,i,L}
     X(j+m)=Pi_i(j,2*(i-1));%Last m elements excluding. All pi_{j,i,H}
end
X=[Pi_0(i+1) X];%Including pi_{0,i}. Note that we take i+1 since 0,0 is included
```

Bibliography

[1] A. Zhang. (2006). A Priority Queueing Analysis of the Short Message Service for GPRS Networks. 3B Co-op Work Report, University of Waterloo.

[2] M.F. Neuts. (1981). Matrix-geometric Solutions in Stochastic Models: An Algorithmic Approach. John Hopkins University Press, Baltimore.

[3] S.M. Ross. (2007). Introduction to Probability Models, 9th Edition. Academic Press, San Diego.

[4] S. Drekic. (2004). Phase-type distributions. In: Encyclopedia of Actuarial Science, Volume 3, J.L. Teugels and B. Sundt (Editors), John Wiley & Sons, Chichester, pp. 1288-1290.

[5] W. J. Stewart. (2009). Probability Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton University Press.