Department of Mathematics and Statistics

Transient Analysis of M/G/1 Queueing Models: Lattice Path Approach

Isnandar Slamet

This thesis is presented for the Degree of

Doctor of Philosophy

of

Curtin University

August 2013

Declaration

To the best of my knowledge and belief, this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: ____________________

Date: 02/08/2013

Acknowledgements

I would like to express my great gratitude to my supervisor, Dr. Ritu Gupta, who has always encouraged and guided me during these years of research. I am very fortunate to have the honour of being her student here in the Department. With her wisdom, knowledge, skills and ability, I sincerely believe that she will be able to benefit many more young researchers. It is always my impression that she is in good health being engaged in the precious area of research. I pray that she always stays in the best health.

Specials thanks to Dr. Narasimaha Achuthan, my co-Supervisor for numerous assistances during my study. His constant guidance and encouragement throughout my PhD program are very beneficial. I do pray that he will always be blessed with wisdom and happiness.

I would like to thank Dr. Roger Collinson as my associate-Supervisor for great assistance, especially his hard work towards the R program carried out in this work. Many times he has to spent hours for discussing the program.

Thanks to Prof. Kanwar Sen for stimulating discussions when he visited the Department in 2010. His presence was a real booster and encouraged me to develop new ideas and kept motivated throughout. Thanks also to Prof. Manju Agarwal for discussion on the preliminary R programming.

Moreover, I would like to thank to the Department of Mathematics and Statistics and Curtin University for providing me a financial support to attend and present my research papers at Australian Statistics Conference 2010, at South Easth Asia Mathematics Society (SEAMS) - Gadjah Mada University (GMU) International Conference 2011, and at World Congress of Statistics and Probability 2012.

Also, I am very thankful to Indonesian Government for the entire financial support on my study.

I also would like to thank to my friends, colleagues and staff in the Department of Mathematics and Statistics for providing a warm, friendly and stimulating research environment during my study here.

Thanks to my wife Yuli Pratiwi who always helps, supports and makes dua for me on every single issue during this tough time. Thanks for taking care of our parents and children.

Finally, my deepest gratitude goes to my parents, brothers, sisters and children (Atikah, Yusa, Shofi, Miqdad and Salman) for their love and consistent support. Without their encouragement and love, I could not possibly complete such a challenge as this study.

Abstract

In this thesis, we develop the explicit expression for pure incomplete busy period (PIBP) density function for $M/G/1$ queueing systems and for incomplete busy period (IBP) density function for $M/G/1$ queueing systems operating under $(0, k)$ and $(k^{'}, k)$ control policies. Under $(0, k)$ control policy, the server goes on the vacation when the system becomes empty and re-opens for service immediately at the arrival of the k^{th} customer. Under $(k^{'}, k)$ control policy, the server starts serving only when the number of customers in the queue becomes k and remains busy as long as there are at least $k^{'}$ customers waiting for service. The explicit form of the incomplete busy period density and other measures of the system performance are not known.

Our approach is to approximate general service time with Coxian 2-phase distribution and represent the queuing process as a lattice path by recording the state of the system at the point of transitions. Herein an arrival into the system is represented by a horizontal step and departure by a vertical step and shift from phase 1 to phase 2 by a diagonal step. Incomplete busy period can then be represented as lattice path starting from $(k_0, 0)$ to (m, n), $m > n$ remaining below the barrier $Y = X$. Control policies imposes additional restrictions on the barrier. Next we use the lattice path combinatorics to count the feasible number of paths and corresponding probabilities.

The above leads to the required density function that has simple probabilistic structure and can be computed using R. In this thesis, we also present the challenges in computing the density using R and illustrate the code and the results.

Keywords: Pure incomplete busy period, Incomplete busy period, $(0, k)$ control policy, $(k^{'}, k)$ control policy, Lattice path, Coxian distribution.

List of Publications Related to This Thesis

1. Isnandar Slamet, Ritu Gupta and Narasimaha R. Achuthan, Roger Collinson, Analysis of Incomplete Busy Period of $M/G/1$ Queues under $(0, k)$ Control Policy, World Congress of Statistics and Probability, Istanbul, Turkey, 2012.

2. Isnandar Slamet, Ritu Gupta and Narasimaha R. Achuthan, Incomplete Busy Period Analysis of $M/G/1$ Queues: A Special Case, Second International Conference on Computing, New Delhi, India, 2011.

3. Isnandar Slamet, Ritu Gupta and Narasimaha R. Achuthan, Lattice Path Approach for Incomplete Busy Periods Analysis of $M/G/1$ Queues Using C_2 Coxian Distribution, The 6^{th} SEAMS-GMU International Conference, Yogyakarta, Indonesia, 2011.

4. Isnandar Slamet, Ritu Gupta and Narasimaha R. Achuthan, Lattice Path Approach for Transient Analysis of $M/G/1$ Queues under $(0, k)$ Control Policies Using C_2 Coxian Distribution, Proceedings of the Seventh Australian Conference on Australian Statistics Association (ASC), Fremantle, Australia, 2010.

Abbreviations

AR	Arrival run
DR	Departure run
FIFO	First in first out
IBP	Incomplete busy period
PIBP	Pure incomplete busy period
LPs	Lattice paths
LPC	Lattice path combinatorics
MTSF	Mean Time to System Failure
SLP	Skeleton Lattice Path
SP1	Structural properties 1
SP1LDE	SP1 for last departure event
SP1LDgE	SP1 in which the last diagonal event happened preceded by a departure
SP2	Structural properties 2
SP2LAEP1	SP2 ending with last arrival event occurred during phase 1 of service
SP2LAEP2	SP2 with last arrival event occurred during phase 2 of service
SP2LDgE	SP2 with the last diagonal at the end of the path and preceded by an arrival
SP3	Structural properties 3
SPLAE	Structural properties for last arrival event
SPLAEV	Structural properties of a LP ending with a last arrival event occurring under vacation period
SPLAEB	Structural properties of a LP that ends with a last arrival event during the busy period

SPLAEB-I	SPLAEB when last arrival event happened in phase 1 of service
SPLAEB-II	SPLAEB when a last arrival event happened in phase 2 of service
SPLDE	Structural properties for last departure event
SPLDgE	Structural properties for last diagonal event

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Background

Queueing system has attracted attention of researchers since 1909 when Erlang analyzed telephone traffic congestion service problem of the Copenhagen Automatic Telephone System as a queueing system (Gross and Harris [37]). Since then, many researchers have successfully modeled and investigated queueing systems arising from a variety of application areas (Panwar et al. [74], Towsley [100], Brunell and Wuyts [14], Kleinrock [52, 53], Buzacott and Shantikumar [15], Marshall and Zenga [62]) such as computer and data systems, communication networks , machine interference, job-shop type systems, telecommunication traffic, semi conductor manufacturing, transportation systems, road traffic, telephone call centers, manufacturing and health care systems.

By queueing systems we refer to a scheme of customers arriving at a system to seek service. Customers could range from phone call to emails, whereas corresponding service could be answering phone call to processing mail. Classically, any queueing system can be described by characterizing its input and output mechanisms. The key characteristics of queueing systems are described below (Gross and Harris [37]).

(1). Arrival process (A). Arrival process describes the arrival mechanism of customers to the system via a probability distribution. We commonly assume the arrival process follows Poisson distribution which adequately supports many real world situations. The customer may arrive in the system individually or in a group. Further, the customers may balk that is they arrive but do not enter the system, or renege if they leave the system after a time without being served.

(2). The service process (S). We may describe the service mechanism of the customers in the system via a probability distribution. For a customer, the time elapsed from the commencement to completion of service is called service time. We commonly assume service time to have exponential distribution. The customers may be served individually or in batches.

(3). Number of servers (n). The number of servers represents the number of service facilities available to serve customers. There may be one or more service facilities.

Based on the above characteristics, queueing systems can be classified by the following convention: $A/S/n$, where A is the arrival process, S is the service process and n is the number of servers.

Other important parameters in the description of a queueing system are the system capacity, size of customer population and the queue discipline. The system capacity indicates the maximum number of customers that can, at any time, be in the queueing system including service facility and in the queue. Customer population indicates the number of customers in the calling population. Finally, queue discipline indicates the manner in which customers are selected for service when a queue has been formed, for example first come-first served (FCFS) and last come-first served (LCFS).

A queueing process with notation $M/G/1$ represents a single server, system where the arrivals are generated by a Poisson process, customers are served as per order of arrival (FCFS) and the service times of customers follow independent and identical general distribution.

Figure 1.1 presents an example of the queueing system in a bank institution. The system includes: 1. Arrival process: this process starts with the arrival of customers. 2. The service process: in this process service is given to a customer.

Figure 1.1: The illustration of queueing system

3. Number of servers: the number of servers represents the number of tellers available to serve customers.

The arrival process and the service process for this system can be investigated to compute performance measures like peak service time. We can study sensitivity of the parameters of the system to increase productivity and reschedule staff.

System performances can be described by measures such as the average number of customers in the system, average number of customers in the queue and the mean waiting time in the system. System performance has been explored for various queueing system by a number of authors. For example the joint distribution of the number of customers serviced during a busy period and the length of the busy period under a $M/G/1$ process has been derived by Rao [78]. Perry et al. [77] obtained closed-form expressions for the Laplace transforms of the lengths of the busy periods of two models of $M/G/1$ and $G/M/1$ type queueing systems with restricted accessibility. Doshi and Heffes [27] considered various queue disciplines FCFS and LCFS of $M/M/1$ queue when discussing the main function of a processor overload control.

Traditionally the performance of a queuing system is investigated after it attains a steady state condition. For several examples of such steady state analysis of general queueing systems, we refer to Takagi [94, 95], Neuts [72, 73], Gross and

Harris [37], Kleinrock [52, 53] and Buzacott and Shantikumar [15].

In real life, there are queueing systems that do not continuously operate for enough length of time and so they do not attain steady state situations. Nevertheless analysis of the performance of such queueing systems in their transient state is equally important to assist managerial decisions regarding optimal use of resources.

Though several queueing scenarios arising from the areas of communication, manufacture and computer systems remain in transient state during their operation time, the research effort in analyzing such systems is considerably lower than the effort on the systems that attain steady state situations.

As noted by Gross and Harris [37], the mathematical analysis of transient systems tends to be complex and thus demands innovative new approaches to analyze them.

Grassman [36] presents a comprehensive account of early research on transient analysis (including the methods available to find transient solutions which include Runge-Kutta, Liou's method and randomization). The transient analysis for Markovian queues (Markovian arrival and service process) has also been extensively explored (Krinik [54], Narahari and Viswanadham [71]).

The study of queueing systems with general distribution is challenging and has received significant attention of researchers. Kleinrock [52, 53] discussed several examples of these $M/G/1$ queueing systems and their evaluation. These queueing applications include modelling manufacturing systems (Buzacott and Shanthikumar [15]), production line (Lee and Hong [57], Vidalis and Papadopulis [101]) and survival times in healthcare models (Marshall and Zenga [62]).

General service time distribution could include probability distributions like exponential, Erlang, hyperexponential, Pareto, phase type and Coxian k-phase. An innovation in tackling general service distribution is to approximate it by a Coxian phase-type distribution (Cox [24], Khosgoftar and Perros [51], Agarwal et al. [1, 2], Harris et al. [41]). This approximation retains Markovian structure leading to simplistic assumption for subsequent analysis.

In many systems, to optimally manage the service time, the server takes vacation from main system when it becomes empty (Takagi [94, 95, 96], Fiems et al. [34], Doshi [26]). Such systems are encountered in areas such as digital communication, computer network and production/inventory systems (Doshi [26], Fiems and Bruneel [33]).

However, majority of the modelings for such system are either restricted to steady state forms or transient solutions are in the form of complex Laplace transforms.

The non-availability of any closed form transient solutions for non-Markovian queues up to 1996 was noted by Borkakaty et al. [10, 11]. The available solutions till date were either in terms of Laplace-Stieltjes transforms (LSTs) or other integrals, so that inversion was required to get the probabilities of interest. For general service distributions and complex queuing scenarios, even with the current computing power, the inversions are complicated and intractable. Avoiding such problems, one can use the lattice path combinatorics (LPC) to study queuing processes. The system probabilities obtained in this case are in explicit closed forms that are straight forward to compute (Borkakaty et al. [10, 11], Agarwal et al. [1, 2], Sen and Agarwal [85]).

LPC starts with representing the behaviour of the queueing process through a sequence of steps represented as lattice path. For example, observing the system at points of transitions, a lattice path can be constructed by representing an arrival into the system by a horizontal step and a departure by a vertical step. In this case, the system size at any point is the difference between the number of horizontal and vertical steps, or the distance between the end point of the path and the barrier $Y = X$. An example of a LP is given in Figure 1.2.

Recently lattice paths approach has been applied extensively in analysing queueing systems (Markovian, general arrival/service), see for example Agarwal et al. [1, 2], Borkakarty et al. [10, 11], Mohanty [68], Sen [82], Sen and Agarwal [85], Sen et al. [89] and Wazalwar and Khaparde [104]. The basic idea is to compute the number of lattice paths that can lead to a given queueing characteristics, like busy period of certain length. Majority of the work for general arrival/service process is restricted to the study of busy period density. See Figure 1.3(a).

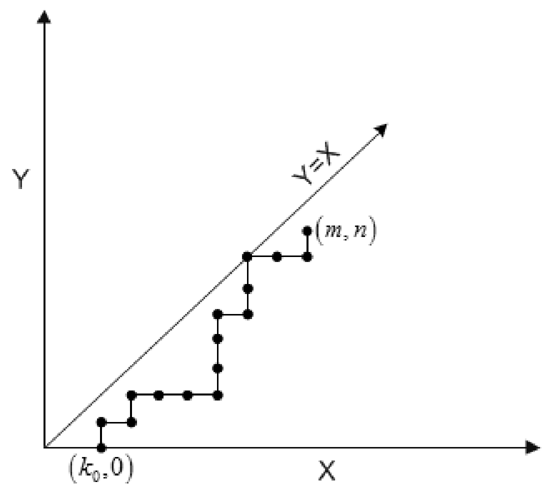


Figure 1.2: An example of a lattice path (LP)

During transient phase, suppose that the queueing system is observed continuously from start time 0 to (current) time t. During this time interval of length t, the system may include alternately several busy and idle periods of the server. This length of time t will be referred to as pure incomplete busy period (PIBP) if it does not include any idle time of server. PIBP can be represented as lattice path starting from $(k_0, 0)$ to (m, n), $m > n$, always remaining below the barrier $Y = X$. See Figure 1.3(b).

Furthermore, the case where a system experiences at least one busy period up to time t will be referred to as incomplete busy period (IBP). IBP can be represented as lattice path starting from $(k_0, 0)$ to (m, n), $m > n$, not crossing (touching or remain below) the barrier $Y = X$. See Figure 1.3(c).

In view of above, in this thesis, we present the study of the transient probabilities of $M/G/1$ queueing systems using lattice path approach. Our approach is to approximate general service time with Coxian 2-phase distribution and represent the queuing process as a lattice path by recording the state of the system at the point of transitions.

We will present the probability density function of the PIBP of $M/G/1$ queues, and the IBP of $M/G/1$ queues operating under control policies. Two control policies considered here are $(0, k)$ and (k', k) control policies. Furthermore, the density functions of PIBP and IBP are computed using R.

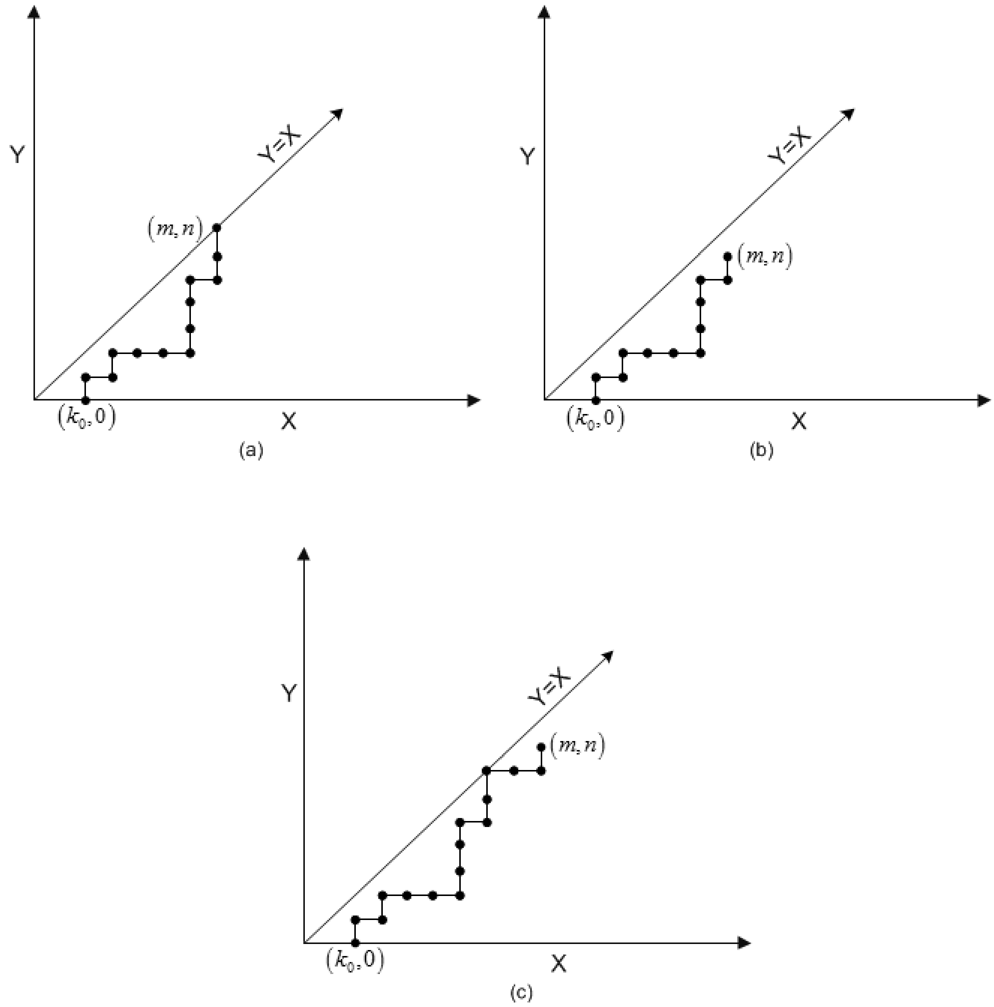


Figure 1.3: An example of a LP: (a). Busy period, (b). Pure incomplete busy period, (c). Incomplete busy period.

This research would lead to significant advancements in the use of the LPC to compute the PIBP of $M/G/1$ queueing systems and the IBP of $M/G/1$ queueing systems operating under control policies. The computation code presented can be used as an input in the optimization process for developing optimal strategies to effectively manage a queueing system.

1.2 Objectives

Although a significant advances in the study of busy period of $M/G/1$ queues and their associated control policies has been made, there are still many problems that require further investigation.

This research presents the probability density function of the incomplete busy period of $M/G/1$ queues and such systems operating under control policy $((0,k),(k^{'},k))$ using lattice path approach. Key element of our methodology is to represent general service distribution by Coxian 2-phase distribution.

Our study of $M/G/1$ queues mainly comprises of three different components. These are: Firstly, represent PIBP for $M/G/1$ queues via a lattice path representations, Secondly, derive the expression for its density and Finally, develop code for numerically computing the density. Next, we derive the IBP density function of $M/G/1$ queues under $(0,k)$ control policy and under $(k^{'},k)$ control policy, following same approach.

Completion of the main objectives would involve the following:

(1). Developing a complete set of structural properties for lattice path representations of PIBP and IBP for $M/G/1$ model. This includes defining the basic notation and terminology, representing the incomplete busy period via lattice path, constructing the run, inserting the diagonals, and approximating the service time by Coxian 2-phase distribution.

(2). Developing the techniques for enumeration of the number of lattice paths corresponding to its structural properties.

(3). Deriving the PIBP density function of $M/G/1$ queues, the IBP density function of $M/G/1$ queues under $(0,k)$ control policy and the IBP density function of $M/G/1$ queues under $(k^{'},k)$ control policy.

(4). Developing computer codes to estimate the busy period densities for above models. The required algorithms and the computational procedures are developed in R computing system [97]. This includes generating all possible lattice paths using the library AlgDesign, filtering the paths satisfying the structural properties of the process and computing corresponding density for each set.

(5). Study the sensitivity of the parameters.

1.3 Outline of the Thesis

In this thesis, we develop various theoretical results for the density of the PIBP and IBP of $M/G/1$ queues by approximating general service by C_2, Coxian 2-phase distribution. The IBP system are also investigated under two control policies namely $(0, k)$ and $(k^{'}, k)$.

The thesis is organized as follows.

- In Chapter 2, we review previous research results relevant to incomplete busy period density of $M/G/1$ queues.
- In Chapter 3, we develop probability density function (pdf) of PIBP of $M/G/1$ queues using lattice path approach and C_2 Coxian distribution.
- In Chapter 4, we develop pdf of IBP of $M/G/1$ queues under $(0, k)$ control policy using lattice path approach and C_2, Coxian 2-phase distribution.
- In Chapter 5, we develop pdf of IBP of $M/G/1$ queues under $(k^{'}, k)$ control policy using lattice path approach and C_2, Coxian 2-phase distribution.

Furthermore, for each model discussed in Chapters 3-5, we develop the corresponding code in R to estimate the pdf of PIBP and IBP for a given t.

- In Chapter 6, we present key finding and present direction for future research. We also present the acknowledgement and references.

Chapter 2

LITERATURE REVIEW

2.1 An Overview

Queueing model is a mathematical representation of the activities in a system that provides a service. Queueing systems could arise from a variety of scenarios: a queue at a bank counter, a queue at a supermarket checkout, a queue of jobs at a computer service and so on. Queueing theory plays an important role in managing an optimal usage of allocating resources to provide efficient service. Broadly queueing theory falls under the class of applied stochastic process identified by the system flow (Kleinrock [52]).

In working with queueing theory we must, first of all, identify the particular real-world system of interest, study this system, and create (or simply choose from the list of models in queueing theory) a mathematical model to represent it (Gross and Harris [37], Kleinrock [52, 53], Larson and Odoni [56], Takagi [94, 95]). Through the analysis of this mathematical model, we then obtain the answers relating to system occupancy/efficiency, which supposedly apply to the original system as well.

During the past hundred years or so, many researchers have focused on investigating several real-world queueing systems with attempts to improve their efficiency. This has led to the formulation of large volume of analytical/computational results for queueing models.

We find that the study on queueing systems is mainly dominated by the assumption that arrival times of customers are independent and identically distributed (i.i.d.). The main reason for this assumption is that in a majority of queueing process observed, the interarrival times or the service times or (ideally) both have exponential distribution. Gross and Harris [37] and Larson and Odoni [56] indicate that most of the queueing systems arising from urban service systems have exponential inter-arrival distribution and hence Poisson process is commonly assumed.

However, the same can not be assumed for service time that are regarded to follow general distribution. Over the past decade, $M/G/1$ queues have been of interest to researchers, especially the busy period analysis and study of control policies.

Table 2.1 presents an overview of the application of the $M/G/1$ queues. We observe that majority of the studies are restricted to the development of models under steady state. Furthermore, control policies, namely T policy and N policy (equivalent to $(0, k)$ policy) results are available under steady state alone. When the transient analysis has been attempted, explicit results are available for busy period density alone. In this thesis, we will focus on $M/G/1$ queues, and provide explicit formula/computation codes for probability density function $[f(.)]$ for pure incomplete busy period (PIBP) for $M/G/1$ queues. Analogous results will be developed for incomplete busy period (IBP) for $M/G/1$ queues operating under control policies.

Table 2.1: Research on $M/G/1$ from different features

Authors	**Service Distribution**	**Analysis**	**Control Policy**	**Features**
Mohammadi and Salehi-Rad (2012) [67]	Truncated normal distribution	Steady state	No	The mean system size, the mean busy period, the probability of a idle period.
Wazalwar and Khaparde (2011) [104]	Coxian 2-phase, C_2 distribution.	Transient	$(0, k)$	Busy period density function.
Liu et al. (2010) [60]	An independent and identically distributed (*iid.*) random variables G with a general distribution function $G(t),\ t \geq 0$.	Steady state	N-policy	The probability generating function of system size distribution.
Zhang and Xu (2010) [109]	An independent and identically distributed (*iid.*) random variables H with general distribution function $H(t),\ t \geq 0$	Steady state	No	Confidence interval for traffic intensity, the mean waiting time in queue, the mean response time, and the mean number of customers in the system.
Choudhury and Tadj (2009) [21]	$B_1(x)$: pdf. of a first phase of regular service and $B_2(x)$: pdf. of a second phase of optional service.	Steady state	No	Busy period distribution and waiting time distribution.
Agarwal et al. (2007) [2]	Coxian 3-phase , C_3 distribution.	Transient	No	Busy period density function.

Chu and Ke (2006) [22]	Empirical bootstrap distribution.	Steady state	No	Bootstrap confidence interval for mean response time.
Hansen and Pitts (2006) [40]	Empirical distribution.	Steady state	No	Non-parametric inference of the service time distribution and the traffic intensity.
Rodrigo (2006) [79]	An independent and identically distributed (*iid.*) random variables B with a general distribution function $B(x)$, $x \geq 0$ and moments $\beta_i = E[X^i]$.	Steady state	No	The estimation of the retrial parameter for constant and non-constant retrial policies.
Ausin et al. (2004) [4]	Phase type distributions.	Steady state	No	The stationary queue size, waiting time and busy period distributions
Fearnhead (2004) [31]	General distribution.	Steady state	No	Likelihood recursion for the queues based on inter-departure time data.
Tadj (2003) [91]	General nonexhaustive service.	Steady state	T-policy	An ergodicity condition and steady state probabilities for the system size at a service completion epoch.
Sen (2002) [82]	Coxian distributions.	Transient	No	Transient analysis of busy period density function.

Jain (2000) [42]	An independent and identically distributed (*iid.*) random variables X_i with a general distribution function $B(x)$, $x \geq 0$ and moments $\beta_i = E[X^i]$.	Steady state	No	The estimate of relative efficiency of the parameters.
Perry et al. (2000) [77]	S_n: the time required for a full service of the n^{th} customer.	Steady state	No	Laplace transform of the lengths of the busy periods.
Wang and Ke (2000) [103]	The service time is *(iid)* random variables having a distribution $S(u),\ u \geq 0$ and a probability density function $S(u),\ u \geq 0$ and mean service time s_1.	Steady state	No	The steady state probability distribution of the number of customers in a finite system.
Bingham and Pitts (1999) [9]	General distribution.	Steady state	No	Construct estimators of arrival rate and of the service time moment generating function.
Gross and Harris (1998) [37]	The service times X *iid* random variables with a general distribution function $B(x)$, $x \geq 0$ and moments $\beta_i = E[X^i]$.	Steady state	No	Mean queue length and the number in system

Insua et al. (1998) [45]	Erlang and Hyper-exponential. distributions	Steady state	No	Bayesian analysis.
Medhi (1998) [64]	The service times X *iid* random variables with distribution function $B(x),\ x \geq 0$ and moments $\beta_i = E[X^i]$.	Steady state	No	Generalizations and extensions.
Sen and Agarwal (1997) [84]	Coxian phase-type distribution.	Transient	No	Busy period density function.
Li et al. (1997) [59]	The service times $\{X_n,\ n \geq 1\}$ are *iid* random variables with distribution $G(x)$, mean $1/\mu$ and hazard rate $\mu(x)$.	Transient	No	Transient solution for the number of the customers at time t, the system availability, failure frequency, mean time failure.
Takagi (1991) [94]	Renewal points.	Steady state	No	Principal performance measures.
Neuts (1989) [73]	The service times X *iid* random variables with distribution function $B(x),\ x \geq 0$ and moments $\beta_i = E[X^i]$.	Steady state	No	Mean queue length and the number in system.
Hernandez-Lerma and Marcus (1984) [43]	Service times for class i customers are independent and identically distributed as a nonnegative random variable S_i with distribution function G_i.	Steady state	No	An optimal adaptive policy.

Machihara (1984) [61]	Hyperexponential distribution	Steady state	No	Traffic estimate errors for the delay system.
Hernandez-Lerma and Marcus (1983) [42]	The service rate is u, the service time of the customer is a random variable with distribution $G(t, u)$.	Transient	No	The determination of optimal adaptive policy.
Thiagarajan and Harris (1979) [98]	Exponential.	Steady state	No	Goodness of t test for the service times
Kleinrock (1975) [52]	Inter-arrival time distribution is exponential.	Steady state	No	Mean queue length and the number in system.
Heyman (1968) [44]	General distribution.	Steady state	T-policy	Obtain the optimal value of T and the optimal cost rate for the minimum cost-rate criterion.

In this chapter, we provide a comprehensive review of $M/G/1$ queueing systems. The rest of the chapter is organized as follows. In Section 2.2, we give a brief description of the steady state analysis. Section 2.3 deals with the transient analysis. Section 2.4 presents critical overview on the methods for analysing queueing systems. Next, we present control policy in Section 2.5. In addition, we discuss the Coxian phase-type distributions in Section 2.6. Finally, lattice path approach is discussed in Section 2.7.

2.2 The Steady State Analysis

As mentioned in Section 2.1, the performances of queueing systems is commonly investigated under the steady state condition. This condition is attained when the system has been in operation for sufficiently long period of time such that the system performance is independent of the initial conditions.

The main reasons for assuming steady state in analyzing a queueing system (Narahari and Viswanadham [71]) are as follows.

1. There are computationally efficient and simple methods for steady-state analysis. For example, the computation of steady-state probabilities in a Markov chain is carried out by solving a system of linear equations.

2. Major results in queueing theory, such as Littles law and Jacksons theorem are all concerned with steady-state analysis.

3. Developments in aggregation and decomposition methods for solving large Markov chain models or large queueing models have also focused on steady-state analysis

Studies on $M/G/1$ queuing systems have been carried out both in continuous-time as well as in their discrete-time counterparts. The later types of models are of interest due to developments of computers and communications wherein the process is naturally discrete.

In the following two subsections, we present an overview of the results for the steady state of $M/G/1$ queueing system operating under continuous and discrete time. Under the continuous time case, we review the transition probabilities, the Pollaczek-Khintchine formula, the steady state distribution of the number in the system and the distribution of waiting time. Under the discrete time case, the mathematical model is presented followed by the key result and the probability distribution function of system occupancy at a random slot.

2.2.1 The $M/G/1$ continuous time queueing system

For $M/G/1$ model, Gross and Harris [37] pointed that a Chapman-Kolmogorov analysis is not possible since general service will lead to the violation of Markovian assumption. Thus, to model this system, we not only need to know what state the system is in at time t, but also how long it has been there.

There have been several approaches in the literature for computing the steady state probability distribution. Gross and Harris [37], and Medhi [64] used difference-differential equations include Bessel functions to analyze the steady state of the $M/G/1$ queueing systems.

The supplementary variable technique was introduced by Cox [46], and has been widely applied to $M/G/1$ queueing systems by Takacs [92], Cohen [23], and others. Based on this supplementary variable technique, Gupta and Rao [38] provided a recursive method to compute the steady state probability distribution of the number of failed machines in the $M/G/1$ machine repair problem with no spares and the cold-standby $M/G/1$ machine repair problem. Further, supplementary variable and Laplace transform techniques are used to solve the $M/G/1$ queueing systems (Choudhury and Tadj [21]). This approach can also be used to study the finite source model as applied to the machine repair problem (Choudhury and Tadj [21]).

2.2.1.1 The Pollaczek-Khintchine formula

Consider an $M/G/1$ queueing system where probability density function of the general service time is denoted by $B(t)$. Let $B(t)$ have finite first two moments denoted by $E(s)$ and $E(s^2)$, respectively.

The offered load, ρ, ($\rho < 1$) for such a system is defined as $\rho = \lambda E(s)$. ρ can be interpreted as long-run fraction of time the server is busy.

Let $E(W)$ be the long-run average time spent per customer in the queue.

$$E(W) = \frac{\lambda E(s^2)}{2(1-\rho)}. \tag{2.1}$$

This equation is often called Pollaczek-Khintchine (PK) formula.

Consequently, mean sojourn time, $E(T)$ is given by

$$E(T) = E(s) + E(W) = \rho + \frac{\rho^2 + \lambda^2 Var(s)}{2(1-\rho)}. \tag{2.2}$$

Further, let $CV^2 = \frac{Var(s)}{E^2(s)}$ then

$$E(W) = \tfrac{1+CV^2}{2}\tfrac{\rho}{1-\rho}E(s)$$

and

$$E(T) = (1 + \tfrac{1+CV^2}{2}\tfrac{\rho}{1-\rho}).$$

By applying Little's formula, one can note that

$$E(N_q) = \lambda E(W)$$

and

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

where N_q and N are the long-run number of customers in the queue and system, respectively.

From PK formula it is easy to note that the mean value for waiting time and sojourn time depends only on the first two moments of the service time. The above Pollaczek-Khintchine (PK) form of the equations can also be expressed in the form of Laplace transforms (Kleinrock [52, 53]).

2.2.1.2 The steady state distribution of number in the system

We refer to Kleinrock [52, 53] for the complete derivation. The z-transform for the number of customers in the system can be expressed as

$$P(z) = \frac{(1-\rho)(1-z)B^*(\lambda - \lambda z)}{B^*(\lambda - \lambda z) - z}. \tag{2.3}$$

This formula is referred to as first form of the Pollaczek-Khintchine (PK) transform equation.

Kleinrock [52, 53] noted that if we can find the Laplace transform of the probability density function for the service time, then we will be able to find $P(z)$ after finding $\Phi(z)$ i.e the probability generating function of the number of customers arrive during the service time of customer n. The coefficient of z^n, i.e. π_n will be given by the power series of $P(z)$. The mean and variance of the number in the system at the moment of service completion can be calculated as we find first and second derivatives $P(1)$.

2.2.1.3 The distribution of waiting time

The time spent in the system by the customers is defined as the total time that the customers spent in queueing and in service until they leave the system. We denote this as random variable Y with probability density function $f(y)$. This material of discussion follows that of Kleinrock [52, 53]. Equation (2.4) expresses the Laplace transform of the distribution of total time spent in the system.

$$F^*(s) = \frac{(1-\rho)sB^*(s)}{s-\lambda+\lambda B^*(s)}. \tag{2.4}$$

This formula is referred to as the second Pollaczek-Khintchine transform equation.

The third Pollaczek-Khintchine equation can be found by considering steady-state conditions and deriving the Laplace transform of the queueing time distribution. Suppose we denote T as the time spent in the queue, and $Y = T + S$ where S is the service time. We note that S is independent of T.

The Laplace transform of the queueing time distribution can be found as follows.

$$F^*(s) = V^*(s)B^*(s). \tag{2.5}$$

This will give us

$$V^*(s) = \frac{(1-\rho)s}{s-\lambda+\lambda B^*(s)}. \tag{2.6}$$

This formula is referred as the third Pollaczek-Khintchine.

2.2.1.4 Constructing a Markov process

For non-Markovian processes, the transition probability can be derived as suggested by Kendall (1953), as in White et al. [105] by observing the system at points where the Markov property holds. These points are referred to as renewal points, that is, t is a renewal point if and only if the future states of the system depend solely on the state at time t. Here, we may note that in a Markov process, certainly, every point is a renewal point. Thus, by considering the system only at renewal points, we are in essence imbedding a Markov process into a non-Markovian environment. This methodology is also referred to as imbedded Markov process (Kleinrock [52, 53]).

In $M/G/1$ queueing systems, immediately after the customer leaves the system, either a waiting customer will enter service station to receive service or the system becomes empty. Thus, the future state solely depends on the number in the present. Those instances at which customers finish receiving service and then leave the system are the renewal points.

2.2.2 The $M/G/1$ discrete time queueing system

Discrete-time queueing systems were introduced by Meisling in 1958 (Woodward [107]). Since then, these discrete queues have been explored leading to the advancements both on the theoretical framework and applications. In past decade or so, the developments of computers and communications in the practical worlds are becoming more and more digital in which the basic unit time (we call it a slot) is finite size and multiple events can take place in this interval (Kleinrock [52]). Furthermore, discrete-time system is characterized by time-slot and synchronous service (Bruneel and Kim [13]).

Based on the fact that computer communication networks is digital, discrete-time approach is appropriate to develop a performance model. Due to its rapid development, the study of discrete queues have become important as queues in such applications are discrete by nature.

Typically in these processes, digital information arrives into a system and enters the buffer storage for certain period of time before being transmitted to its destination. Before transmission takes place, the information is chopped into fixed length packets and is synchronized to the availability of the clock signals, which are generated at regular interval by a system clock (slots). The service constitutes of the transmission of packets with service time being an integer number of slots envisaged for transmitting the package.

The system transition occurs only at the end of each slot. Hence the packet (customer) can not leave the system at the end of a slot in which it arrived. At best, the service (transmission) of such a packet can occur at the next slot, after which the packet can leave the system. This kind of service is what distinguishes the discrete-time system from the continuous-time counterparts.

There have been interest on associated optimization problems arising from the prediction of the behaviour of communication networks for computers and other sources of digital information. Some common questions to be answered are what should be the size of buffer capacity without buffer overflow, the network protocol gives the best delay, the busy periods, and others related output characteristics (Woodward [107]). Study on these fundamental performance measures has been carried out in various scientific disciplines which can be classified into Computers, Communication Systems and Networks, Civil and Transportation Engineering (Kleinrock [52]).

In telecommunication industries, recent focus has been on broadband integrated services digital network (BISDN) for its capability of providing a common interface for future communication needs including video, data and speech. Since information in BISDN is transported by means of discrete unit of the asynchronous transfer mode (ATM) cell, this has increased focus study on discrete-time systems (Bruneel and Kim [13]).

In communication systems and networks, considerable efforts have been made in the past regarding the use of queueing models for many different applications. Discrete-time queueing models are commonly used to analyze and understand the performance of a variety of communications systems such as statistical multiplexer, polled systems and broadcast systems (Towsley [100]), ATM switching elements, traffic concentrators (Brunell and Wuyts [14]), asynchronous time division multiplexing (ATDM) systems and integrated digital voice-data with synchronous time division multiplexing (STDM) for voice sources systems (Kekre et al. [50] and Rodrigo and Falin [80]). Wittevrongel and Bruneel [12] and Bruneel [12] introduced the performance evaluation of ATM multiplexer and switcher.

Discrete queues are commonly analyzed by setting the corresponding system of

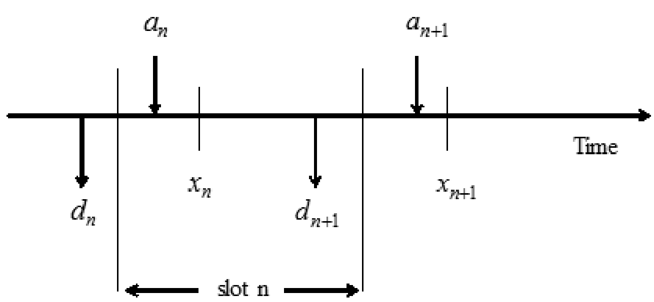


Figure 2.1: Notations for arrivals, departures and state of a discrete-time queueing system

equation and solving these via mathematical techniques involving Laplace transform. Therefore the Laplace-Stieltjes transform and the results of discrete-time queueing systems are extensively used here to quantify system performance. The lattice path frame work as discussed later in Section 2.7 provides a convenient and natural framework for the study of discrete queues.

2.2.2.1 Discrete-time queueing notations

We denote the number of customers (cells) entering the system and the number of customers (cells) departing during slot n as $\{a_n, n = 0, 1, 2...\}$ and $\{d_{n+1}, n = 0, 1, 2, ...\}$, respectively with $d_0 = 0$. It is important to note that $\{a_n\}$ is a sequence of i.i.d random variables with a specified distribution. The queue length process at the slot boundary is denoted by $\{y_n, n = 0, 1, 2...\}$ with y_0 arbitrary. Hence, the following state-equation can be established.

$$y_{n+1} = y_n + a_k - d_{n+1}. \tag{2.7}$$

We denote the queue length process observed just after the customer entering the system during slot n as $\{x_n, n = 0, 1, 2...\}$ where $x_n = y_n + a_n$. We may graph this conventions as in Figure 2.1. (Figure taken from Woodward [107])

2.2.2.2 The fundamental result

We consider general discrete-time queueing model with single server, infinite waiting room, the number of arrivals follows general probability distribution. Let $a(n)$

be the probability density function of arrival process, i.e. P[n customers arrive during one slot], $n \geq 0$. The corresponding probability generating function is indicated as

$$A(z) = \sum_{n=0}^{\infty} a(n)z^n.$$

The service times of the customers are assumed to constitute a set of iid. positive random variables with common mass function $t(n)$= P[service of a customer takes n slots], $n \geq 1$ and corresponding probability generating function

$$T(z) = \sum_{n=0}^{\infty} t(n)z^n.$$

Let us define $P_k(x, z)$ as the generating function of joint probability of the state vector (h_k, u_k) at the beginning of slot k, where h_k is the remaining number of slots needed to service the customer currently in service at the beginning of slot k and u_k is the system occupancy at the beginning of slot k.

Hence, we have

$$\begin{aligned} P_k(xz) &= E[x^{h_k} z^{u_k}] \\ &= \sum_{i=0}^{\infty}\sum_{n=0}^{\infty} Pr[h_k = i, u_k = n]x^i z^n. \end{aligned}$$

Take into account all possible (h_k, u_k), we obtain

$$\begin{aligned} P_{k+1}(xz) &= E[x^{h_{k+1}} z^{u_{k+1}}] \\ &= A(0)[1 - T(x)]\{P_k(0, 0) + R_k(0, 0)\} \\ &+ A(z)\frac{xT(x) - 1}{x}P_k(0, 0) \\ &+ A(z)[T(x) - z]R_k(z) \\ &+ \frac{A(z)}{x}P_k(x, z), \end{aligned}$$

where

$$R_k(z) \cong \sum_{n=1}^{\infty} Pr[h_k = 1, u_k = n]z^{n-1}. \tag{2.8}$$

For a system that attains steady-state conditions, we obtained Equation (2.9) below. Complete derivation can be found in (Bruneel [12]). This equation is the fundamental result from which we can derive the performance measures.

$$P(xz) = [1 - A^{'}(1)]S^{'}(1)]\{1 - xz\frac{[1 - A(Z)][T(X) - S(A(z))]}{[x - A(z)][z - S(A(z))]}\}. \tag{2.9}$$

2.2.2.3 The system occupancy at random slot

In order to study the system occupancy under steady-state conditions, the following steady state random variables are considered :

u : system occupancy at the beginning of a random slot,

d : system occupancy at customer departure times,

c : system occupancy as seen by new arrivals.

The respective probability generating functions are denoted by $U(z), D(z)$ and $C(z)$.

The probability generating functions of the system occupancy at random slot when steady state probability distribution has been attained can be derived from Equation (2.9) and is given below. See Bruneel [12] for the complete steps.

$$U(z) = P(1, z) = [1 - A'(1)]S'(1)]\{\frac{[z-1][S(A(z))]}{[z - S(z)]}\}. \tag{2.10}$$

2.3 The Transient Analysis

In many situations, practitioners need to know the information on the behavior of the system up to time t, when the system has not yet attained steady state conditions. Transient analysis is appropriate (Narahari and Viswanadham [71]) in applications such as manufacturing systems, systems with failure states, unstable queueing systems, and systems with fluctuating or non-stationary workloads. Transient analysis is important for studying queueing measures such as performance over finite intervals, sensitivity analysis, first passage time computation, settling time computation, and deriving the behavior of models as they approach equilibrium.

Literature on transient analysis of Markov chain models spans many inter-disciplinary areas. Few closed-form expressions exist for the transient and the nonstationary behavior of queueing systems. For a good survey on early research on transient analysis, see Grassman [36].

Let $P_{ij:k}(t)$ denote the probability that starting with k units at time $t = 0$, there are i arrivals and j departures in the system up to time t. Clearly $i + k - j \geq 0$. Let $P_n(t)$ be the probability that there are n units in the system at time t. Then,

$$P_n(t) = \sum_{i=n-k}^{\infty} P_{i,i+k-n;k}(t).$$

To enable computation of range of performance measures in transient analysis of queueing systems, the event of interest is the pair of cumulative number of arrivals and that of departures, realized up at time t.

The transient analysis of queueng models is targetted towards computing the above probability. Queueing applications where transient analysis is important include modelling manufacturing systems (Buzacott and Shanthikumar [15], Kleinrock [52, 53]), production line (Lee and Hong [57], Vidalis and Papadopulis [101]) and survival times in healthcare models (Marshall and Zenga [62]). In these applications, computation of system performance is of interest.

In general, the approaches to obtain the transient probabilities of interest can be broadly classified into two categories, namely solving system equations and direct combinatorial methods. Each of these approaches is described below.

The general solution of the $M/M/c$ queue has been studied by Saaty [81] and Parthasarathy and Sharafali [76]. The former derived the Laplace transform of $P_n(t)$ and expressions for the two-server queue. The latter derived explicit expressions when $c = 1, 2, 3$ and 4. The solution of the relevant difference differential equations for such a system in terms of Bessel functions have also been obtained (Gross and Harris [37]).

The combinatorial methods have been used by Mohanty [68], Sen et al. [89], Böhm and Mohanty [7], Sen and Jain [84] and Mohanty and Panny [69] to compute probabilities of interest for Markovian queueing system. The results from combinatorial methods are in the form of explicit expressions that can be computed. Similar results and techniques are also presented by Muto et al. [70] for $M/M/c$ queues using the approach by Towsley [100].

For non-Markovian models, the computation of transient solution is significantly more challenging. Let us consider $M/G/1$ model. In this case arrivals are generated by a Poisson process but the service time is permitted to be a non-exponential distribution. Note that it assumes no explicit distribution for the service time but assume that the service times of all customers are independently and identically distributed (i.i.d). General service time distribution could include probability distributions like exponential, Erlang, hyperexponential, Pareto, phase type and Coxian two-phase etc.

In this model with a single server, the following assumptions are satisfied:

i. Customers arrive following the Poisson distribution with constant rate λ. The probability density function of $A(t)$, the number of arrivals in $(0, t]$, is given by

$$P(A(t) = k) = \frac{e^{-\lambda t}(\lambda t)^k}{k!}, k \geq 0.$$

ii. The service time follows a general distribution with the density function, say $B(t)$.

One of the major performance of a queueing system is the transient distribution of the queue length since other characteristics such as the waiting time can be derived from this quantity. For the purpose of deriving this, we need to develop equations for the system as below.

(i) $Q(t)$ denote the number of customers in the system at time t (including the one under service).

(ii) $\xi(t)$ as an indicator variable which is 1 if the server is busy at time t or otherwise 0.

(iii) Transition probability $X_m(t) = m + A(t) - D(t)$, where $D(t)$ is the number of service completions during $(0, t]$.

Finally, deriving the joint distribution of $Q(t)$ and $\xi(t)$ facilitated by $X_m(t)$ will lead to the result, i.e. the transient distribution of the queue length.

A major breakthrough in computation of transient probabilities for $M/G/1$ is by approximating general service by Coxian-phase type distribution (Cox [24]). Approximation by phase type distribution leads to Markovian set up for analysis.

2.4 Comparisons of Methods for Analyzing Queueing Systems

Based on the discussions in the above sections, it can be noted that, in most cases, the only approach available for obtaining probability distribution for queueing performance measures is through the use of a combination of transform analysis and numerical analysis techniques.

As a matter of fact due to the availability of computing power in modern era, some of these approaches are very powerful. They apply to quite general queueing systems and provide important information about the queueing phenomena that occur, while requiring only a minimum amount of knowledge about the characteristics of interarrival times, service times, queue discipline, and so on. Nevertheless, the final results are obtained from a numerical black box and are not transparent, on the other hand, combinatorial framework provides elegant framework for understanding probabilistic structure of the results which will eventually lead to computational efficiency.

In general, the intensive research of queueing theory can be summarized as below (Larson and Odoni [56]).

(1) Almost all the existing important results of queueing theory are obtained for equilibrium conditions (i.e., with the queueing system operating in the steady state, in engineering parlance).

(2) Even assuming equilibrium conditions, queueing theory runs into enormous mathematical difficulties in all but relatively few types of situations.

(3) Quite often, the choice facing an analyst is between, on the one hand, using a realistic mathematical model for which almost no results can be obtained and, on the other, using a simplified model that provides results of questionable validity for the problem at hand.

2.5 Control Policy

Many real life queueing systems are operated under control policies determined heuristically with the objective of optimal management of the system. As a simple example, a server serving at a bank teller may continue to serve the customers till no more customers are waiting. At this point the server may switch to other tasks (i.e. take vacation) like counting cash and return back to service as soon as a new customer arrives. In this case we say that the queueing system is operating under server vacation and a number of different rules (control policies) can be set for the server to return to the main systems.

A comprehensive survey of queueing systems under vacations is presented by Doshi [26]. Server vacations may be induced due to lack of work, server failure, or another task being assigned to the server. Such scenarios occur commonly in applications like computer maintenance and testing, preventive maintenance jobs in a production system, priority queues, etc. Doshi [26] also elaborated on many applications of various vacation models.

In most simple case, when a server becomes idle, he leaves for a vacation. At the end of a vacation period, service commences if a customer is present in the queue. Otherwise, the server takes another vacation immediately and continues in the same manner taking multiple vacations until he finds at least one customer waiting upon returning from a vacation.

Kao and Narayanan [48] have discussed the $M/M/n$ queue with multiple vacations of the servers using a matrix-geometric approach. They proposed algorithms for computing the stationary queue length distribution, the expected length of busy period, the mean waiting time, and the waiting time distribution.

Levy and Yechiali [58] studied $M/M/n$ queue where any server goes on vacation whenever there are no customers waiting in the system at a service completion. The server returns to the system at the end of vacation and takes another vacation if it finds empty queue. Using a transform approach and under the steady state conditions, they derived the distribution of the number of busy servers and the mean queue length of the queue.

Kumar and Madheswari [55] studied a $M/M/2$ Markovian queue with two servers and multiple vacations using a matrix-geometric method. They considered different service rate for the two servers, and computed the stationary queue length distribution and mean system size.

Chae and Kim [18] considered the busy period in the $GI/M/1$ queue with multiple exponential vacations. They derived the transform of the joint distribution of the length of a busy period, the number of customers served during the busy period, and the residual interarrival time at the instant the busy period ends for the $GI/M/1/MEV$ queue, where MEV stands for multiple exponential vacations.

Techniques of regenerative point processes have been used by Sridharan and Mohanadivu [90] to obtain various measures of system effectiveness and thereby maximizing the profit. The time-dependent, steady-state system availability, reliability, mean time to system failure (MTSF), and profit function are obtained numerically.

The performances of $M/G/1$ queueing systems under vacations have been studied by Choudury and Tadj [20]. They considered these systems when there is second phase of optional service subject to breakdowns occurring randomly at any instant while serving the customers. The probability distribution function of steady state of queue size has been derived as well as busy period distribution and waiting time distribution.

Garg [35] have investigated busy period analysis for two-unit standby systems, using regenerative point technique. They obtained busy period analysis, availability analysis, and MTSF.

One other type of control policy is $(0, k)$ control policy. Under this set up, the server goes on the vacation when the system becomes empty and re-opens for service immediately at the arrival of the k^{th} customer. This policy without vacations was initially studied by Yadin and Naor [108].

Chae and Lim [17], based on Takacs, presented a procedure to obtain the joint transform of the length of a busy period, the number of customers served during the busy period, and the remaining interarrival time at the instant the busy period ends for the N-policy $GI/M/c$ queue.

Sen et al. [89] used the combinatorial methods of lattice path for deriving transient solution of $M/M/1$ under $(0, k)$ control policy. Their approach was to first discretize the interval $(0, t)$ by dividing the time interval into t/h time slots of length h each and then represent the corresponding queue process by a lattice path. Associating the probabilities with the lattice paths and passing on to the limit as $h \to 0$, the probabilities of interest are computed.

An extension of $(0, k)$ control policy is $(k^{'}, k)$ control policy. Under this set up, the server starts serving only when the number of customers in the queue becomes k and remains busy as long as there are at least $k^{'}$ customers waiting for service. Under this policy, Böhm and Mohanty [7] and Sen and Gupta [88] analysed $M/M/1$ queueing systems. Ke et al. [49] considered this policy for $M/M/c$ queue.

Other types of control policies explored are T and D control policies. Under T policy, server takes multiple vacations each of duration T when the system becomes empty. The vacation stop when the server finds a non empty system on return at the end of vacation. The T control policy has been considered by Sen and Gupta [86, 87] under $M/M/1$ queues. Likewise, transient analysis was conducted under the same policy for $M^b/M/1$ queues (Sen and Gupta [88]).

Under D policy, when the system is being exhausted, then server takes vacation and starts service only when the total units or workload crosses D, i.e. $D \geq 0$. Analysis of bulk queues operating under D-policy with multiple vacations has been done by Agarwal and Dshalalow [3] using fluctuation method.

Study of general queueing system under control policies is still a challenge. For example, no explicit results are available for $M/G/1$ queues under control policy. These results can be developed using LPC and will be presented in this thesis.

2.6 The Coxian Phase-type Distributions

The Coxian phase-type distributions, characterized by parameters μ and λ describes duration until an event occurs in terms of a process consisting of k latent phases, leading to the Markovian structure. These distributions represent the time to absorption for a finite state Markov chain in continuous time, generalise the exponential distribution and provide a flexible and useful modelling tool (McGrory et al. [63]).

A phase-type distribution describes a Markov process, $\{X(t); t \geq 0\}$, say, where the system moves through some or all of k transient states, or phases, before moving to a single absorbing state $k+1$ (McGrory et al. [63]). See Neuts [32] for a full description. The phases are governed by the transition probabilities (McGrory et al. [63]).

$$P\{X(t+\delta t) = j+1|X(t) = j\} = \alpha_j \delta t + o(\delta t), j = 1, \ldots, k-1$$
$$P\{X(t+\delta t) = k+1|X(t) = j\} = \beta_j \delta t + o(\delta t), j = 1, \ldots, k$$

Here δt represents a small time increment. The $\{\alpha_j\}$ are the transition rates between the transient states and the $\{\beta_j\}$ describe the transition from any of the transient phases to the absorbing state.

In the Coxian phase-type model (see Cox and Miller [25]) the system starts in the first phase and then moves through the transient phases sequentially before eventually being absorbed from any one of them. See Figure 2.2 for an illustration.

This Coxian phase-type distribution is a type of Markov model that describes duration until an event occurs in terms of a process consisting of k of latent phases.
Figure 2.2 presents the Coxian k-phase approximation, where μ_i is the average rate of serving the customers in phase i and β_i is the probability that a customer will depart the system after completing the phase i for $i : 1, \ldots, k$. The α_j is the probability that a customer will move from phase j to phase $j+1$ for $j : 1, \ldots, (k-1)$. Hence, $\alpha_j + \beta_j = 1$ for $j : 1, \ldots, (k-1)$, and $\beta_k = 1$.

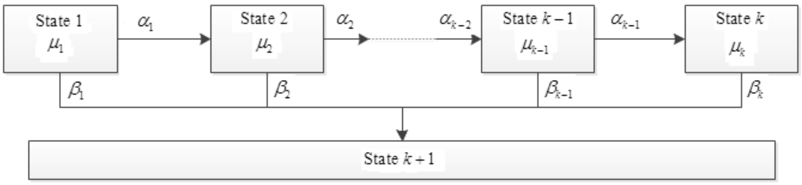


Figure 2.2: Coxian k-phase distribution

Coxian phase-type distribution has become popular since it possess the Markovian property (Iversen and Nielsen [46]) and a wide class of distributions can be very closely approximated by a Coxian phase-type distribution (Cox [24]). Cox [24] demonstrated that any distribution with rational Laplace-Stieltjes transform (LST) and square coefficient of variation (CV^2) lying in $[1/k, \infty)$ can be approximated by a sequence of k fictitious independent stages; see also Harris et al. [41]. The case where k phases of service are exponentially distributed is the so called Coxian k-phase, C_k distribution. The general distribution with $CV^2 > 1$, has been well approximated by Coxian 2-phase (C_2) by Khosgoftaar and Perros [51].

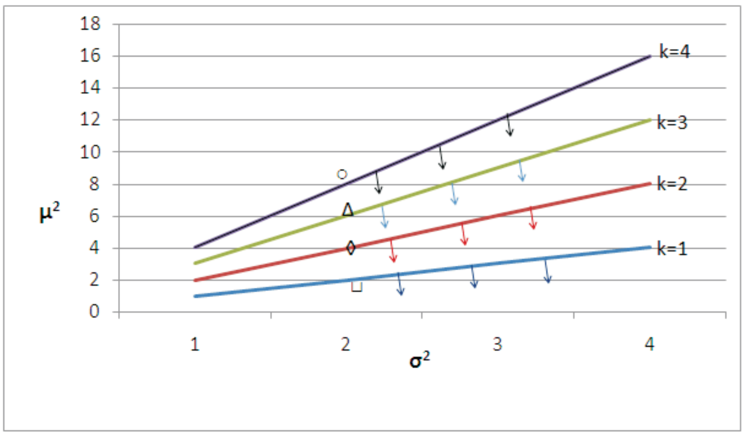


Figure 2.3: The relation between CV^2 and k

Figure 2.3 presents all the possibilities of the values of CV^2 versus k. When the CV^2 falls in region $\ominus$ then the general distribution can be approximated by C_k for $k \geq 4$ distributions. When the CV^2 falls in region $\triangle$ then the general distribution can be approximated by C_k for $k \geq 3$ distributions. When the CV^2

falls in region $\diamondsuit$ then the general distribution can be approximated by C_k for $k \geq 2$ distributions. When the CV^2 falls in region $\Box$ then the general distribution can be approximated by C_k for $k \geq 1$ distributions.

Recently, Sen [82], Sen and Agarwal [85], Agarwal et al. [1], Borkakarty et al. [11, 10] and Wazalwar and Khaparde [104] utilized C_2 distribution as approximations of $M/G/1/n$, $C_2/M/1$, $G/G/1$, C_2/C_{2b} /1, $GI/M/1/n$ and $M/G/1$ type of queueing models, respectively. Their analysis is based on LP approach.

On approximating the general distribution by 3-phase Cox distribution C_3, LP approach has been used for transient analysis of non-Markovian $M/G/1$ queues by Agarwal et al. [2].

Approximating general service by Coxian phase-type falls under the approach 3 (see page 29), with a valid approximation. This aspect will be adopted in this thesis.

2.7 Lattice Path Combinatorics (LPC)

Lattice path comprises of a path in (x, y) plane moving along lattice from point, say, (x, y) to (x_1, y_1). Counting the number of possible paths from (x, y) to (x_1, y_1) under boundary restriction is referred to as lattice path combinatorics and has been of interest for a long time. In addition to queueing theory investigated in this study, such lattice paths are used in many applications. For example, lattice path was applied to generalize h factorial polynomial of order n (Buzeatenou and Dumocos [16]. In the area of computerized data storage and retrieval, the lattice path counting techniques was applied to the analysis of a direct-access storage and retrieval scheme known as linear probing by Mendelson [65]. The relationship between lattice path enumeration and tennis ball problem has been elaborated by Fallon et al. [29]. Ericksen [28] studied lattice path combinatorics under the Delannoy identities.

The application of lattice path models in queuing theory lead to the evolution of new results. For instance the book by Fayolle et al. [30] presents interesting results on random walks in quarter planes. Further Banderier and Flajolet [5] developed enumerative lattice path results under various constraints i.e. bridges, excursions and meanders using the kernel method. Bŏhm [6] applied lattice path counting techniques to study the time-dependent distribution of $M^a/M^b/1$ queues and open tandem queues with and without global blocking and demonstrated the kernel method alternative to arrive at these results.

An important result in LPC is Ballot theorem (Takacs [93], Mohanty [68]). This theorem proposes the probability of the event that during the counting of the votes, the number of votes for A is at all times greater than the number of votes for B. Given the total number of voters is μ. Candidate A obtains n votes and is elected; candidate B obtains $m = \mu - n$ votes. The probability of this event is given by $(2n - \mu)/\mu = (n - m)/(n + m)$. Based on this theorem, batch Ballot theorem was established by Mercankosk et al. [66].

The Ballot theorem was, initially, applied to count the number of a lattice path from $(k, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$. A number of results on counting these paths are available in Takacs [93], Feller [32] and Mohanty [68]. Mercankosk et al. [66] derived the first passage distribution matrix, G, for the general $M/G/1$ queue through the batch Ballot theorem.

Combinatorial techniques have been used for queueing systems by Takacs [93], Mohanty [68]. Using lattice path counting technique, Mohanty and Panny [69] and Sen and Jain [83] derived transient solution of $M/M/1$ queues. The combination of lattice path counting and combinatorial techniques has been used by Sen at el. [84] and Böhm and Mohanty [7] to derive the transient solution to $M/M/1$ model with $(0, k)$ and $(k^{'}, k)$ control policies, respectively.

The approach used by Sen et al. [89], Böhm and Mohanty [7], Sen and Jain [41], and Sen and Gupta [87] during (up until 2009) comprised of the following steps: divide the time interval $(0, t)$ into a sequence of t/h subintervals (time slots) each of length $h(> 0)$; define the random variable X_i equal to $1, -1$ or 0 respectively if the i^{th} subinterval has an arrival, a departure or neither an arrival nor a departure (stay). A lattice path stipulated by a queueing model can be constructed via plotting realizations of X_i on the lattice path, by representing arrival, departure and stay by unit horizontal, vertical and diagonal step.

Next, the number of possible paths corresponding to the queueing model are counted for a given event. This counting is done in two stages. First, need to delete all stays. This results in skeleton path. Then, all possible skeleton paths can be counted, for example using Reflection Principle and First Passage Principle (Mohanty [68]). Further, the stays deleted in the first stage of the procedure are inserted at the vertices of the skeleton path by using Balls into Cells Technique (Feller [32]) to generate the corresponding lattice paths envisaged.

Next, the transition probabilities associated with the queueing systems are established. Using the result on counting paths, the probabilities for such discrete queueing systems can be computed. Finally, the transient probabilities is computed by taking the limit as $h \to 0$ (Sen at el. [84], Böhm and Mohanty [7]). This process gives the results for both discrete and continuous time case. However, the limiting process is not always easy to compute.

Moving away from the approach of developing transient probability as a limiting form of discrete model, Agarwal et al. [1, 2] proposed observing the process at time of transitions. They conducted the analysis by first defining time points at (like in the study of embedded Markov process) which transitions take place. They further proposed approximating general service/arrival process by Coxian phase-type distribution. This provided Markovian structure for analysis. Next, the lattice path was constructed by considering the sequence of arrivals, departures and stay (or movement to next phase). The sequence then can be represented by the movement of a point along the lattice path in the XY plane. A lattice path will be transformed into skeleton path by removing the diagonal steps. Using the so called run definition, the total number of lattice paths can be calculated. Finally, the probability density function of events of interest can be derived.

This method has been successfully used to compute busy period density functions of various queues with single server using C_2, Coxian 2-phase type distribution as an approximation of either general arrival process or service times or both as in Sen [85]. Using the same approach, Sen and Agarwal [85] derived the transient queue length distribution of $C_2/M/1$ queues and the busy period analysis of bulk queues $C_b/M/1$. Further, they obtained the busy period density of $M/G/1$ queues using C_3, Coxian 3-phase distribution and the busy period density of $GI_b/G/1$ queues using C_2, Coxian 2-phase distribution (Agarwal et al. [1, 2]). Borkakarty et al. [10, 11] have studied busy period analysis of $C_2/C_{2b}/1$ queue. They have also used the lattice path approach for busy period density of $G/G/1$ queues using Coxian 2-phase distributions. Wazalwar and Khaparde [104] have derived busy period density function for $M/C_2/1$ queues under $(0, k)$ control policy where arrival and service process are correlated.

The other notable work in this area is by Muto et al. [70] where lattice paths counting method is applied to analyze the transient $M/M/c$ queueing systems. The closed-form solution for the probability of exactly i arrivals and j departures within a time interval of length t has been derived. The derivation of the probability is based on the counting of paths from the origin to (i, j) on the XY-plane, that have exactly x-steps whose depth from the line $Y = X$ is d ($d = 0$, 1,. . . , c - 1). The concept of depth is analogue to the run as defined by Agarwal et al. [2].

LPC, therefore, serve as elegant and useful analysis tools in a variety of application areas (Mohanty [68]). This method will be further explored in this study.

2.8 Summary

This chapter presents a comprehensive review of $M/G/1$ queueing systems. Key challenges in study of $M/G/1$ systems are

(1) Under steady state conditions, the probability distribution for queueing performance measures are commonly obtained via a combination of transform analysis and numerical analysis techniques.

(2) Steady state solutions for $M/G/1$ queueing systems have been obtained for both the continuous time as well as in their discrete-time counterparts.

(3) The two approaches for obtaining the transient probabilities of $M/G/1$ queueing systems are solving system equations and direct combinatorial methods. The former method is computationally intense while later is mathematically complex.

(4) A breakthrough in computation of transient probabilities for $M/G/1$ queueing systems is through approximating general service time distribution by Coxian phase-type distribution. This process retains Markovian structure and a wide class of distributions can be closely approximated by a Coxian phase-type distribution.

(5) Once the general service time is approximated by Coxian phase-type distribution, lattice path approach can be used to compute transient probabilities of various queuing measures and that for queues operating under control policies.

(6) Study of general queueing system under vacations has been attempted however explicit results are not available for $M/G/1$ queueing systems under $(0, k)$ and (k', k) control policies.

Chapter 3

M/G/1 QUEUE

3.1 General

In recent years, research focus in queueing models has been on developing the methods to compute performance measures of non-Markovian queueing systems. Performance measures include the length of busy period, pure incomplete busy period (PIBP) and incomplete busy period (IBP).

As we stated in (2.4), until 1995, a vast majority of transient results available for non-Markovian queues were either in terms of Laplace-Stieltjes transforms (LSTs) or other integral transforms that are much complicated, intractable and hard to implement.

Therefore, lattice path technique has been developed to provide transient solutions in explicit form. Some authors have successfully derived the transient solutions for Markovian queueing system for $M/M/1$ (Sen and Jain [83], Bohm and Mohanty [7]), and for $M^b/M/1$ (Sen and Gupta [88]).

For non-Markovian queues, general service/arrival can be approximated using C_2, Coxian 2-phase distribution. This approach has been used to derive busy period density function for $M/G/1$ (Sen and Agarwal [85, 84]), $G/G/1$, $\mathrm{GI}_b/\mathrm{G}/1$ queues (Agarwal et al. [1]), $G/G/1$ queues (Borkakarty et al. [11]). Busy period density and queue length distributions of $M/G/1$ queues approximating general distribution by C_3, Coxian 3-phase distribution has been derived by Borkakaty et al. [11].

Explicit form of the probability density function of the PIBP of $M/G/1$ queues using LP approach are not known. Therefore, in this Chapter, we would mainly study and compute the closed form solution for such systems. Motivated by the previous work, we approximate general service distribution by C_2, Coxian 2-phase and study the PIBP of $M/G/1$ ($\backsim M/C_2/1$) model using the LP approach.

The rest of the chapter is organized as follows. In the following section, we recapitulate the definition of lattice path and briefly explain its application to determine the density function of PIBP of $M/G/1$ queueing system. Section 3.3 presents the $M/C_2/1$ model. Section 3.4 presents the results on counting of paths and subsequent computation of transient probabilities. In section 3.5, we present the numerical computation of density for PIBP for range of parameters. Finally the summary is presented in section 3.6.

3.2 The $M/C_2/1$ Model

We consider a $M/G/1$ queueing model (the Poisson arrival process, general service times distributions and single server) and compute the probability density function of PIBP of this queue through LPC approach by approximating general service distribution by Coxian 2-phase distribution. Hence, we consider a $M/C_2/1$ queueing model. The Coxian distribution as illustrated in Figure 3.1 is a Coxian 2-phase distribution that describes duration until an event occurs in terms of a process consisting of latent phases.

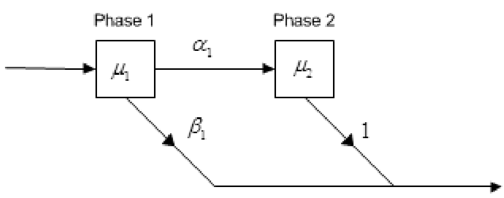


Figure 3.1: Coxian 2-phase distribution

We start with the assumption that the system is non-empty when the system starts. Let k_0 denote the initial number of customer in service. The interarrival time follows exponential distribution and the parameter λ represent the average rate of customers entering the queueing system. Let μ_i be the average rate of serving customers in phase i, $i = 1, 2$. The probability that a customer will move from phase 1 to the phase 2 will be denoted as α_1, likewise the probability that a customer will leave the system completely after completing phase 1 of service will be denoted as β_1, where $\alpha_1 + \beta_1 = 1$. All customers depart after phase 2 of service.

3.2.1 Transitions

For determining the transient solution, we consider time interval $(0, t)$. Observing the system at point of transitions, we let $T_0 = 0$, and $T_1, T_2, \ldots$ be the sequence of time points at which transitions take place. Let $X_0, X_1, X_2, \ldots$ be the number of customers at time points $T_0, T_1, T_2, \ldots$, respectively, where $X_0 = k_0$ from the initial condition. Then $\{X_n\}$ is a Markov chain that has the transition probability matrix Q, where for all $n \geq 0, u = 1, 2$,

$$Q(i,j) = P\{X_{n+1} = j | X_n = i\}$$

$$= \begin{cases} \frac{\lambda}{\lambda+\mu_u}, & \text{an arrival takes place if a customer is undergoing phase u of service} \\ & (j = i+1), u = 1, 2 \\ \frac{\beta_1\mu_1}{\lambda+\mu_1}, & \text{a customer departs after completing phase 1 of service} \\ & (j = i-1) \\ \frac{\mu_2}{\lambda+\mu_2}, & \text{a customer departs after completing phase 2 of service} \\ & (j = i-1) \\ \frac{\alpha_1\mu_1}{\lambda+\mu_1}, & \text{a customer enters phase 2 of service after completing phase 1} \\ & (j = i). \end{cases}$$

The holding time in each state is an exponential random variable with a parameter depending on the state given below:
$P\{T_{n+1} - T_n > t | X_n = i\} = e^{-(\lambda+\mu_u)t}$, if a customer is undergoing phase u of service, $u = 1, 2$.

3.2.2 Lattice path representation of $M/C_2/1$ Model

LPC starts with representing the behaviour of the queueing process at points of transitions through a sequence of steps represented as lattice path. For example a lattice path can be constructed by representing an arrival into the system by a horizontal step and departure by vertical step. In this case the system size at any point is the difference between the number of horizontal and vertical steps, or the distance between the end point of the path and the barrier $Y = X$. Thus arrival of a customer during any phase of service, departure of a customer that can occur at any phase of service and entry into phase 2 will be denoted by a horizontal unit step, a vertical unit step and a diagonal of $\sqrt{2}$ unit step, respectively.

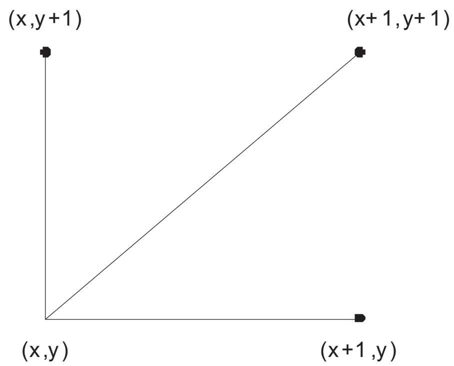


Figure 3.2: A sequence of steps represented as lattice path

Let $(x, y), (x \geq k_0)$ denote a vertex of the LP representing the $M/C_2/1$ queueing process at any point, $(x, y + 1)$ denotes a departure after any phase, $(x + 1, y)$ denotes an arrival during any phase and $(x+1, y+1)$ denotes a customer entering phase 2. The possibility of movements are illustrated in Figure 3.2.

The vertical (horizontal) step will be denoted by a solid line or a dotted line accordingly as departure (arrival) occurs after (during) phase 1 or phase 2 of the service respectively.

As an example, a lattice path representation of $M/C_2/1$ model is given in Figure 3.3. In this example, a LP represents a sequence such that, for $k_0 = 1$, the server is still active at the point $(15, 12)$. Because the number of departures is not equal to the number of arrivals at any point, the lattice path remains below the barrier $Y = X$.

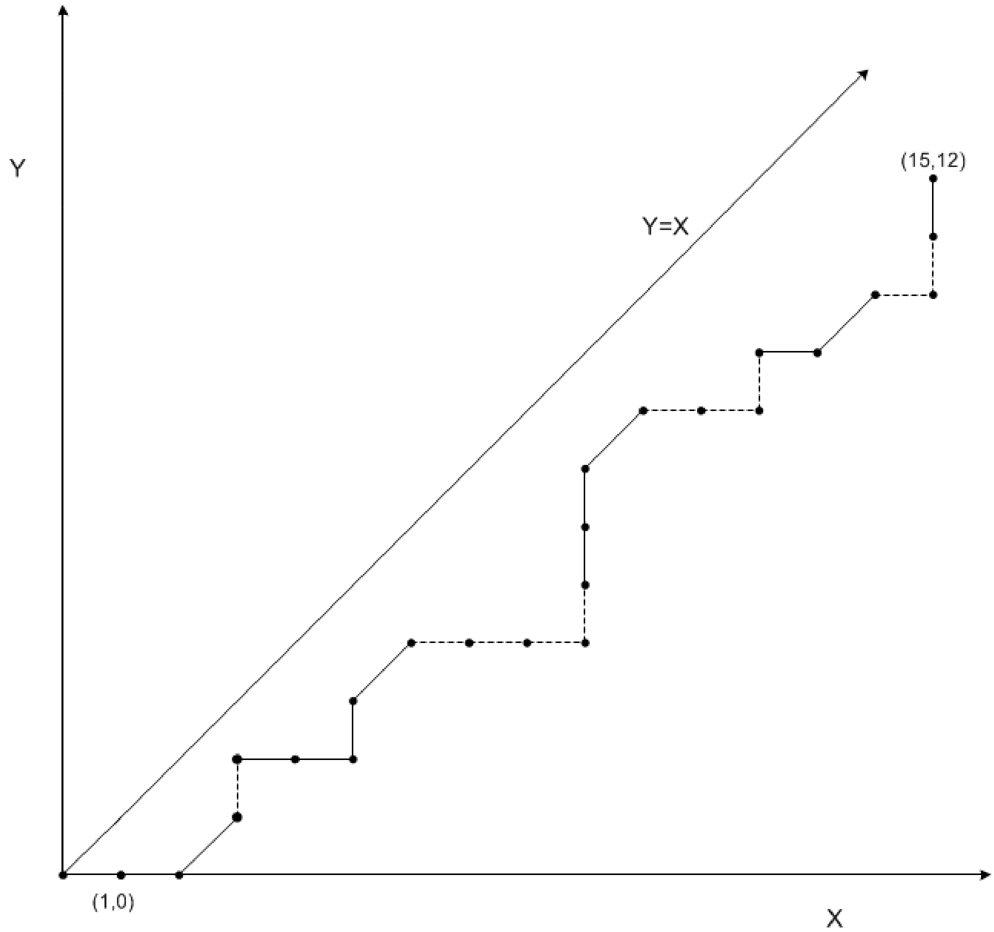


Figure 3.3: $M/C_2/1$ model. Lattice path representation

3.2.3 Counting of lattice paths

For the purpose of counting of LPs, we first transform lattice path (as in Figure 3.3 to a skeleton lattice path (SLP) by removing all diagonals. A skeleton path corresponding to Figure 3.3 is illustrated in Figure 3.4.

Now, it can be seen from Figure 3.4 that this SLP only consist of horizontal and vertical runs which represent arrivals in phase 1 and departures after phase 1, respectively. For SLP we define run as follows.

Definition (Agarwal et al. [2]) Run: A sequence of consecutive horizontal (vertical) steps bounded on each side by a vertical (horizontal) step is called arrivals run denoted by AR (departures run denoted by DR).

The sequence of horizontals starting from the origin and preceding the first vertical step as well as the sequence of verticals at the end following the last horizontal are also called the arrivals run (AR) and departures run (DR), respectively.

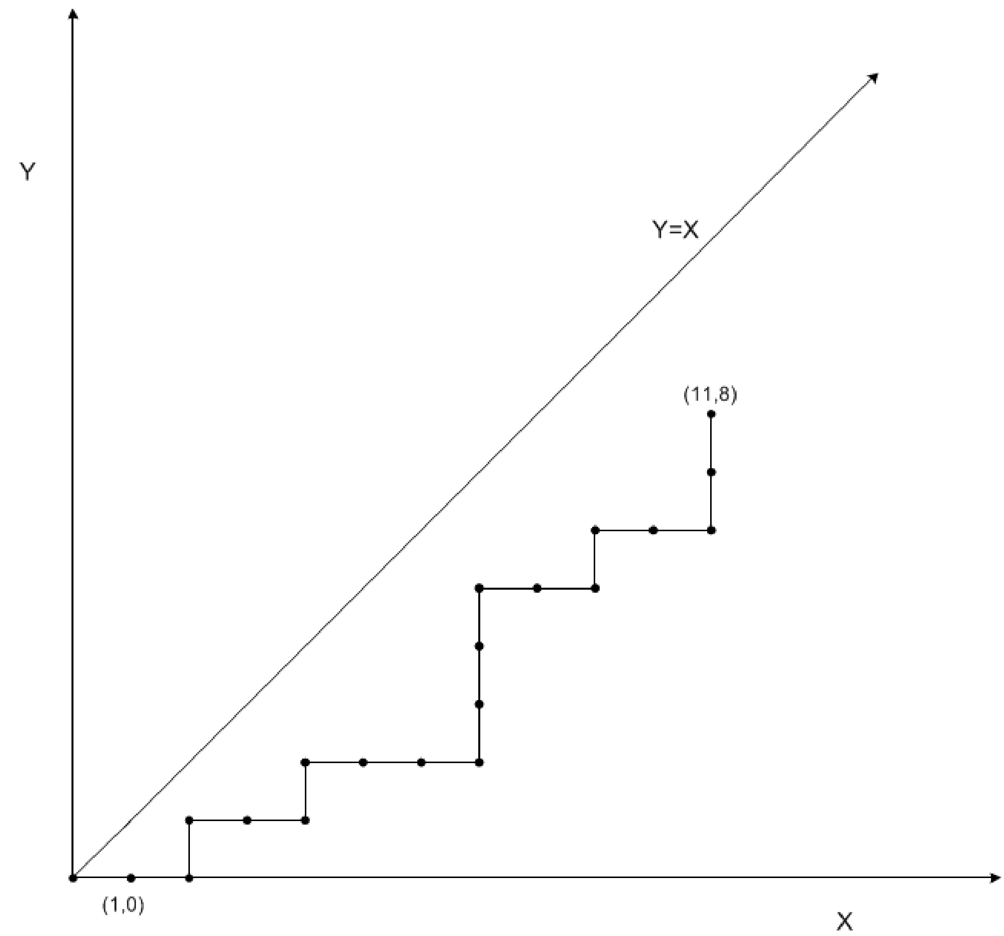


Figure 3.4: Skeleton path of an example of a lattice path representation

Finally, to count the number of LPs from, say $(k_0, 0)$ to (m, n), we also need to consider all possibilities of inserting the diagonals into horizontal and/or vertical runs.

Since we are approximating the service time by C_2, Coxian 2-phase distribution, therefore, while inserting the diagonals, the following restrictions will imply on inserting runs.

- Two or more consecutive diagonal can not appear in any horizontal run or vertical run.
- In a vertical, run any number of diagonals may occur.
- The first vertical step following a diagonal step has to be a dotted vertical step. Horizontal step(s) following a diagonal step and preceding a vertical step are dotted ones.
- Two or more consecutive dotted vertical steps cannot occur.
- A dotted vertical step can not immediately be preceded by a vertical step (departure after phase 2 cannot be preceded by departure after phase 1).
- In any horizontal run only one diagonal can be inserted.

Finally, for a given set of horizontal and vertical runs, we have to count the number of possible LPs that can be generated keeping in view the above restrictions on the insertions of diagonals.

3.2.4 Notation and terminology

Table 3.1 presents the basic notation and terminology that we use in this chapter and in general in the thesis. In most of our work, we follow notation and terminology similar to that used by Agarwal et al. (2007).

To illustrate these notations, we refer to Figure 3.3 where $k_0 = 1, m = 15, n = 12, p = 4, q = 4, m - p - q - k_0 = 6, n - p - q - k_0 = 3, r = 5, l_i : (2, 2, 3, 2, 2), L_i : (1, 1, 3, 1, 2), \underset{\sim}{L} : (2, 2, 3, 2, 2, 1, 1, 3, 1, 2), i : (1, 3, 4, 5), l_{\underset{\sim}{i}} : (2, 3, 2, 2)$ and $p_{\underset{\sim}{i}} : (2, 0, 0, 1)$.

3.3 Pure Incomplete Busy Period (PIBP) of $M/G/1$

This section presents a brief description of the methodology used to determine the pdf of pure incomplete busy period (PIBP) of $M/G/1$ queueing system under

Table 3.1: The notation and terminology

Notation	**Description**
m	Total number of arrivals (horizontal steps) and customers shifting to phase 2 service.
n	Total number of departures (vertical steps) and customers shifting to phase 2 service.
k_0	Initial number of customers at the start of busy period.
r	Number of arrival runs (AR)) which is same as the number of departure runs (DR) ($r \geq 1$).
p	Total number of diagonals inserted in AR and/or DR ($p \geq 0$).
q	Total number of diagonals inserted in AR.
$p-q$	Number of diagonals inserted in DR.
L_i	Length of the i^{th} DR ($i = 1, 2, \ldots, r$), $L_i > 0$.

transient state where PIBP (see page 11) indexed with t and k_0, refers to a continuous period of service till time t, starting with k_0 customers. See section 1.1 and Figure 1.3(b) for further details. Several authors (Sen and Jain [83], Bohm and Mohanty [7], Sen and Gupta [88], Borkakarty et al. [2, 30]) have computed the density functions of busy period of $M/G/1$ queueing system using lattice path combinatorics (LPC) techniques. The proposed methodology approximates a general service distribution by a 2-phase Coxian distribution and extends the LPC technique to compute the pdf of PIBP of $M/C_2/1$ model.

Notation	**Description**
l_i	Length of the i^{th} AR ($i = 1, 2, \ldots, r$) $l_i > 0$.
$\underset{\sim}{L}$	$(l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$.
$\underset{\sim}{L}^*$	$(l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$.
$\underset{\sim}{i}$	$(i_1, i_2, \ldots, i_q)$, index of AR in each of which a diagonal is inserted.
$l_{\underset{\sim}{i}}$	$(l_{i_1}, l_{i_2}, \ldots, l_{i_q})$, lengths of AR with index $\underset{\sim}{i}$
$\underset{\sim i}{p}$	$(p_{i_1}, p_{i_2}, \ldots, p_{i_q})$ distances from extreme left end points where diagonals are inserted in AR runs $\underset{\sim}{i}$.

Let $f_{k_0}(t)$ denote the pdf of PIBP of $M/C_2/1$ for $t \geq 0$, where k_0 is the initial number of customers in the system. The queueing process of $M/C_2/1$ during a PIBP may be represented by a lattice path in a two dimensional plane starting at $(k_0, 0)$ and ending at (m, n). The queueing process is observed at the points of transitions over of time interval $(0, t)$. At this stage, an arrival, a departure or shifting of a customer to phase 2 on the completion of phase 1 service is represented by a unit of horizontal, unit of vertical or $\sqrt{2}$ unit of diagonal steps, respectively.

In the lattice path for $M/G/1$ model, a vertex (x, y) has the following interpretation:

$x =$ the sum of number of arrivals, number of initial customers and number of customers shifting to phase 2 service,

$y =$ the sum of number of departures and number of customers shifting to phase 2 service.

Thus, in the lattice path we have $x > y$, in other words lattice path will never touch the barrier $Y = X$ in the plane.

Thus, a PIBP during $(0, t)$ with k_0 initial customers in the system is represented by a lattice path starting from $(k_0, 0)$ to (m, n), $m > n$, always remaining below the barrier $Y = X$. The pure incomplete busy period of a queueing system with $k_0 = 1$ is represented by lattice path, as illustrated in Figure 3.5. In the graph of the lattice path we represent an arrival or departure occurring during the progress of phase 1 (phase 2) service by solid (dotted) horizontal or solid (dotted) vertical line. Furthermore, a shift to phase 2 service from phase 1 is denoted by solid diagonal line in the graph of LP.

Such a lattice path (corresponding to a PIBP during $(0, t)$ with k_0 initial customers in the system) will either end with a vertical step (departure), a horizontal step (arrival) or a diagonal step (shift to phase 2 service).

The set of all lattice paths satisfying the properties of PIBP period $(0, t)$ is partitioned into three cases as follows:

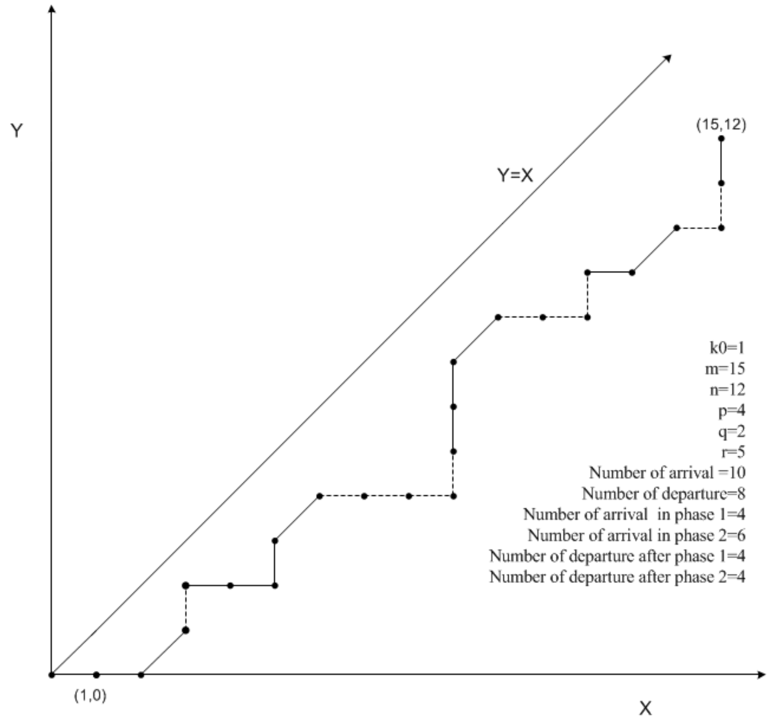


Figure 3.5: Lattice path representation of PIBP of $M/C_2/1$ model.

(i) The set of lattice paths ending with solid (or dashed) vertical line corresponding to a departure after phase 1 (or phase 2) service.

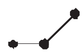
(ii) The set of lattice paths ending with solid (or dashed) horizontal line corresponding to an arrival during phase 1 (or phase 2) service.

(iii) The set of lattice paths ending with a solid diagonal line corresponding to a customer shifting to phase 2 of service. Note that such a solid diagonal line can occur immediately after a solid vertical line (a departure after phase 1 service), dotted vertical line (a departure after phase 2 service) or solid horizontal line (an arrival during the operation of phase 1 service).

Table 3.2 illustrates the decomposition of lattice paths into disjoint groups as explained above in (i)-(iii).

Next, the pdf $f_{k_0}(t)$ is estimated by first counting the number of lattice paths satisfying the properties of PIBP corresponding to the ending structure of the lattice paths (See Table 3.2, column 4). Next, the probabilities corresponding to such paths are computed using transition probabilities corresponding to $M/C_2/1$ model to arrive at $f_{k_0}(t)$.

Table 3.2: Structural properties of LP's ending event and its related theorems

Phase	Events	Representation	Theorem
	- Departure after phase 1 of service		3.4.1
	- Entry into phase 2 of service from phase 1 following departure after phase 1 of service		3.4.4
Phase 1	- Entry into phase 2 of service from phase 1 following departure after phase 2 of service		3.4.4
	- Arrival during phase 1 of service		3.4.2
	- Entry into phase 2 of service from phase 1		3.4.5
	- Departure after phase 2 of service		3.4.1
Phase 2	- Arrival during phase 2 of service		3.4.3

3.4 Results

In this section, first, we develop a complete set of structural properties for different events, namely last departure event, last arrival event and last diagonal event described in Table 3.2. Next, we develop the technique to enumerate the number of lattice paths satisfying specified structural properties. Finally, the expression for the incomplete busy period of $M/G/1$ is derived.

3.4.1 Structural properties of lattice paths

In this section, we present special feature of paths corresponding to the scenario described in Table 3.2 and use them to count the corresponding number of lattice paths.

3.4.1.1 Structural properties 1 (SP1)

Given non-negative integers $k_0, m, n;\ p, q;\ r(r > 1);\ l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_r$; a lattice path from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising p diagonals, $m - p$ horizontal steps, $n - p$ vertical steps and ending with a vertical step. Such a path satisfies the following structural properties:

(a) $m - p$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq k_0, l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - p$,

(b) $n - p$ vertical steps form r runs of lengths $L_1, L_2, \ldots, L_r$, respectively, satisfying $L_1, L_2, \ldots, L_r > 0$ and $\sum_{i=1}^{r} L_i = n - p$,

(c) $l_1 \geq \text{Max}(k_0, L_1 + 1), \sum_{i=1}^{u} l_i > \sum_{i=1}^{u} L_i, u = 1, 2, \ldots, r$,

(d) q diagonals are inserted each in any q out of r horizontal runs (including the vertices at both ends of the runs).

The conditions (a) - (d) above would be referred to as structural properties 1 (SP1) of PIBP.

Theorem 3.4.1. (The number of LPs satisfying SP1 for last departure event (SP1LDE)

For non-negative integers $k_0, m, n;\ p, q;\ r(r > 1);\ l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_r$; let $LP_1\big(k_0, m, n; p, q; r, \underset{\sim}{L}\big)$ denote the number of LPs satisfying the SP1. Furthermore, the remaining $p - q$ diagonals are inserted each at any $n - p - r$ vertices available along the vertical runs.

Then, for $r \geq 1$ and $m > k_0$,

$$LP_1\big(k_0, m, n; p, q; r, \underset{\sim}{L}\big) = \sum_{(R_7, R_8)} \binom{n - p - r}{p - q} \tag{3.1}$$

where

$$R_7 = \Big\{ \big(i_1, i_2, \ldots, i_q\big) \colon 1 \leq i_1 < i_2 < \ldots < i_q \leq r \Big\},$$

$$R_8 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_q}\big) \colon \triangle \leq p_{i_s} \leq l_{i_s},\ s = 1, 2, \ldots, q \Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_s > 1, \\ k_0, & \text{if } i_s = 1. \end{cases}$$

Proof. Before proofing the above theorem, let us consider generic steps taken to prove the theorem. These are

(i) to consider a LP from $(0, 0)$ to (m, n),

(ii) to delete all the diagonal steps,

(iii) to concatenate the steps to form a skeleton path consisting of AR and DR satisfying the properties of respective structural properties,

(iv) to enumerate the number of LP by inserting all possible diagonals.

For the above theorem, let us consider a LP from $(0, 0)$ to (m, n), delete all the diagonal steps, and concatenate the steps to form a skeleton path consisting of AR and DR as described by vector $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$ and satisfying the properties (a) to (c) of SP1 defined above. We now focus on this unique skeleton path and enumerate the number of LP generated by this. This can lead to by inserting diagonals as described in conditions (d).
Let q diagonals be inserted into runs numbered $i_1, i_2, \ldots, i_q$ each of lengths of $l_{i_i}, l_{i_2}, \ldots, l_{i_q}$ at distances $p_{i_i}, p_{i_2}, \ldots, p_{i_q}$ from the extreme left end points. The remaining $p-q$ diagonals will be inserted into any $p-q$ vertices out of $n-p-r$. The number of ways of doing this is $\begin{pmatrix} n-p-r \\ p-q \end{pmatrix}$. Now summing $\begin{pmatrix} n-p-r \\ p-q \end{pmatrix}$ over all possible q-tuples, $(i_1, i_2, \ldots, i_q)$ defined by R_7 and all insertion points $p_{i_i}, p_{i_2}, \ldots, p_{i_q}$ defined by R_8, we get (3.1). □

It is easy to claim the following lemma using Theorem 3.4.1.

Lemma 3.4.1. Let $LP_1(k_0, m, n; p)$ be the number of LP_s from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising of $m - p$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n-p$ vertical steps and p diagonals, ending with a departure event, then summing (3.1) over r, q and L, we find

$$LP_1(k_0, m, n; p) = \sum_{(R_4, R_5, R_6)} LP_1\big(k_0, m, n; p, q; r, \underset{\sim}{L}\big) \tag{3.2}$$

where

$R_4 = \{r : 1 \leq r \leq \max(\min(m - p - k_0 + 1, n - p), 1)\}$,
$R_5 = \{q : \max(0, 2p - n + r) \leq q \leq \min(r, p)\}$,

$$R_6 = \Big\{L : L_1 + 1), l_i \geq \text{Max}(k_0, L_1 + 1), \sum_{i=1}^{u} l_i > \sum_{i=1}^{u} L_i,$$
$$u = 1, 2, \ldots, r, \sum_{i=1}^{r} l_i = m - p, \sum_{i=1}^{r} L_i = n - p\Big\}.$$

3.4.1.2 Structural properties 2 (SP2)

For non-negative integers $k_0, m, n;\ p, q;\ r(r > 1);\ l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_{r-1}$, and $L_r = 0$; a lattice path from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising p diagonals, $m - p$ horizontal steps, $n - p$ vertical steps and ending with a horizontal step. Such a lattice path satisfies the following structural properties:

(a) $m - p$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq k_0, l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - p$,

(b) $n - p$ vertical steps form $r - 1$ runs of lengths $L_1, L_2, \ldots, L_{r-1}$, respectively, satisfying $L_1, L_2, \ldots, L_{r-1} > 0$ and $\sum_{i=1}^{r-1} L_i = n - p$,

(c) $l_1 \geq \text{Max}(k_0, L_1 + 1), \sum_{i=1}^{u} l_i > \sum_{i=1}^{u} L_i, u = 1, 2, \ldots, r - 1$,

(d) q diagonals are inserted each in any q out of r arrival runs (including the vertices at both ends of the runs).

The conditions $(a) - (d)$ above would be referred to as structural properties 2 (SP2) of PIBP.

Theorem 3.4.2. (The number of LPs satisfying SP2 ending with last arrival event occurred during phase 1 of service (SP2LAEP1))
For non-negative integers $k_0, m, n;\ p, q;\ r(r > 1);\ l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_{r-1}$; $\underset{\sim}{L}^* = (l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_{r-1}, 0)$; let $LP_1^*\big(k_0, m, n; p, q; r, \underset{\sim}{L}^*\big)$ denote the number of LPs satisfying the SP2 and ending with the arrival under phase 1 of service. Furthermore the remaining $p - q$ diagonals are inserted each at any $n - p - r + 1$ vertices available along the departure runs.

Then, for $r \geq 1$, $m > k_0$ and $p \geq 0$,

$$LP_1^*\big(k_0, m, n; p, q; r, \underset{\sim}{L}^*\big) = \sum_{(R_7^1, R_8)} \begin{pmatrix} n - p - r + 1 \\ p - q \end{pmatrix}, \tag{3.3}$$

where

$$R_7^1 = \Big\{ \big(i_1, i_2, \ldots, i_q\big) \colon 1 \le i_1 < i_2 < \ldots < i_q < r \Big\},$$

$$R_8 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_q}\big) \colon \triangle \le p_{i_s} \le l_{i_s},\ s = 1, 2, \ldots, q \Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_s > 1, \\ k_0, & \text{if } i_s = 1. \end{cases}$$

Proof. Following generic steps used for proofing Theorem (3.4.1), hence for a lattice path satisfying SP2 and ending with the last arrival under phase 1 of service, the insertion of any diagonal should not take place in last AR. Hence $i_q < r$. To complete the proof which is similar to the proof of Theorem 3.4.1, the number of vertices available where the remaining $p-q$ diagonals can be inserted must be counted. It is obvious that since LP ends with an arrival, $L_r = 0$, and $\sum_{i=1}^{r-1}(L_i) - (r-1) = n-p-r+1$. Hence the total vertices available $n-p-r+1$, along DR.

Therefore,the remaining $p-q$ diagonals will be inserted into any $p-q$ vertices out of $n-p-r+1$. The number of ways of doing this is $\begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix}$.

Now summing $\begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix}$ over all possible q-tuples, $i_1, i_2, \ldots, i_q$ and $p_{i_i}, p_{i_2}, \ldots, p_{i_q}$, we get (3.3). □

Lemma 3.4.2. Let $LP_1^*(k_0, m, n; p)$ the number of LP_s from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising of $m-p$ horizontal steps (including those from $(0,0)$ to $(k_0, 0)$), $n-p$ vertical steps and p diagonals, then summing (3.3) over r, q and L^*, we find

$$LP_1^*(k_0, m, n; p) = \sum_{(R_4, R_5, R_6^*)} LP_1^*\big(k_0, m, n; p, q; r, \underset{\sim}{L}^*\big) \tag{3.4}$$

where

$$R_4 = \{r : 1 \le r \le \max(\min(m-p-k_0+1, n-p), 1)\},$$

$$R_5 = \{q : \max(0, 2p-n+r) \le q \le \min(r, p)\},$$

$$R_6^* = \Big\{ \underset{\sim}{L}^* : l_i \ge \max(k_0, L_1+1), \textstyle\sum_{i=1}^{u} l_i > \sum_{i=1}^{u} L_i, u = 1, 2, \ldots, r-1$$
$$\textstyle\sum_{i=1}^{r} l_i = m-p, \sum_{i=1}^{r-1} L_i = n-p \Big\}.$$

Theorem 3.4.3. (The number of LPs satisfying SP2 with last arrival event occurred during phase 2 of service (SP2LAEP2))
For non-negative integers $k_0, m, n;\ p, q;\ r(r > 1);\ l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_{r-1};$ $\underset{\sim}{L}^* = (l_1, l_2, \ldots, l_r;\ L_1, L_2, \ldots, L_{r-1})$; let $LP_2^*\big(k_0, m, n; p, q; r, \underset{\sim}{L}^*\big)$ denote the number of LPs satisfying the SP2 with the arrival event under phase 2 of service. Furthermore, the remaining $p-q$ diagonals are inserted each at any $n-p-r+1$ vertices available along the departure runs.

Then, for $r \geq 1$, $m > k_0$ and $p \geq 1$,

$$LP_2^*\big(k_0, m, n; p, q; r, \underset{\sim}{L}^*\big) = \sum_{(R_7^2, R_8^1)} \begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix}. \tag{3.5}$$

where

$$R_7^2 = \Big\{ \big(i_1, i_2, \ldots, i_q\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{q-1} < i_q, i_q = r \Big\},$$
$$R_8^1 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_q}\big) \colon \triangle \leq p_{i_s} \leq l_{i_s},\ \ s = 1, 2, \ldots, q-1,\ \text{and}$$
$$0 < p_{i_q} < l_{i_q} \Big\},\quad \triangle = \begin{cases} 0, & \text{if } i_s > 1, \\ k_0, & \text{if } i_s = 1. \end{cases}$$

Proof. Following generic steps used for proofing Theorem (3.4.1), since the last arrival has to occur in phase 2 of service, the insertion of a diagonal out of q diagonals should take place in last AR but not at its right end vertex. Hence $i_q = r$ and $0 < p_{i_q} < l_{i_q}$. The proof then can be completed similar to the proof of Theorem 3.4.2. □

Lemma 3.4.3. Let $LP_2^*(k_0, m, n; p)$ be the number of LP_s from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising of $m-p$ horizontal steps (including those from $(0,0)$ to $(k_0, 0)$), $n-p$ vertical steps and p diagonals, then summing (3.5) over r,q and L^*, we find

$$LP_2^*(k_0, m, n; p) = \sum_{(R_4, R_5, R_6^*)} LP_2^*\big(k_0, m, n; p, q; r, \underset{\sim}{L}^*\big), \tag{3.6}$$

where

$$R_4 = \{r : 1 \leq r \leq \max(\min(m-p-k_0+1, n-p), 1)\},$$
$$R_5 = \{q : \max(0, 2p-n+r) \leq q \leq \min(r, p)\},$$
$$R_6^* = \Big\{ \underset{\sim}{L}^* : l_i \geq \max(k_0, L_1+1), \textstyle\sum_{i=1}^{u} l_i > \sum_{i=1}^{u} L_i, u = 1, 2, \ldots, r-1$$
$$\textstyle\sum_{i=1}^{r} l_i = m-p, \sum_{i=1}^{r-1} L_i = n-p \Big\}.$$

3.4.1.3 Structural properties 3 (SP3)

Structural properties 3 (SP3) of PIBP for counting the number of LPs end on a diagonal preceded by a departure (an arrival) remains same as that of SP1 (SP2). For the purpose of enumerating the number of LPs satisfying structural properties, the event preceding the diagonal need to be considered either as a departure event or as an arrival event.

Theorem 3.4.4. (The number of LPs satisfying SP1 in which the last diagonal event happened preceded by a departure (SP1LDgE))
For non-negative integers k_0, m, n; p, q; $r(r > 1)$; $l_1, l_2, \ldots, l_r$; $L_1, L_2, \ldots, L_r$; $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$, let $LP_2\big(k_0, m, n; p, q; r, \underset{\sim}{L}\big)$ denote the number of LPs satisfying the SP1 in which a last diagonal at the end of lattice path is preceded by a departure event. Furthermore, the remaining $p - q$ diagonals are inserted each at any $n - p - r$ vertices available along the vertical runs.

Then, for $r \geq 1$, $m > k_0$ and $p \geq 1$,

$$LP_2\big(k_0, m, n; p, q; r, \underset{\sim}{L}\big) = \sum_{(R_7, R_8)} \begin{pmatrix} n - p - r \\ p - q - 1 \end{pmatrix} \tag{3.7}$$

where

$$R_7 = \Big\{ \big(i_1, i_2, \ldots, i_q\big) \colon 1 \leq i_1 < i_2 < \ldots < i_q \leq r \Big\},$$

$$R_8 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_q}\big) \colon \triangle \leq p_{i_s} \leq l_{i_s},\ s = 1, 2, \ldots, q \Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_s > 1, \\ k_0, & \text{if } i_s = 1. \end{cases}$$

Proof. Following generic steps used for proofing Theorem (3.4.1), hence out of $(p - q)$ diagonals to be inserted, one diagonal should be inserted at the end of lattice path. Therefore, the remaining $(p - q - 1)$ diagonals have to be inserted in $(n - p - r)$ vertices. Hence the proof is complete. □

Note that the expression in (3.7) can be obtained from the expression in (3.1) by setting $LP_1\big(k_0, m - 1, n - 1; p - 1, q; r, \underset{\sim}{L}\big)$, so that to every LP we attach the last diagonal that takes the path from $(m - 1, n - 1)$ to (m, n).

Lemma 3.4.4. Let $LP_2(k_0, m, n; p)$ be the number of LP_s from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising of $m - p$ horizontal steps (including those from $(0,0)$ to $(k_0,0)$), $n-p$ vertical steps and p diagonals, ending with a diagonal preceded by a departure, then summing (3.7) over r,q and L, we find

$$LP_2(k_0,m,n;p) = \sum_{(R_4,R_5,R_6)} LP_2\big(k_0,m,n;p,q;r,\underset{\sim}{L}\big), \tag{3.8}$$

where

$$R_4 = \{r : 1 \le r \le \max(\min(m-p-k_0+1, n-p), 1)\},$$
$$R_5 = \{q : \max(0, 2p-n+r) \le q \le \min(r,p)\},$$
$$R_6 = \Big\{L : l_i \ge \mathrm{Max}(i_0, L_1+1), \textstyle\sum_{i=1}^{u} l_i > \sum_{i=1}^{u} L_i,$$
$$u = 1,2,\ldots,r,\ \textstyle\sum_{i=1}^{r} l_i = m-p, \sum_{i=1}^{r} L_i = n-p\Big\}.$$

Theorem 3.4.5. (The number of LPs satisfying SP2 with the last diagonal at the end of the path and preceded by an arrival (SP2LDgE)) For non-negative integers $k_0,m,n;\ p,q;\ r(r>1);\ l_1,l_2,\ldots,l_r;\ L_1,L_2,\ldots,L_{r-1}$; $\underset{\sim}{L}^* = (l_1,l_2,\ldots,l_r;L_1,L_2,\ldots,L_{r-1})$, let $LP_3\big(k_0,m,n;p,q;r,\underset{\sim}{L}^*\big)$ denote the number of LPs satisfying the SP2 in which a last diagonal at the end of lattice path is preceded by an arrival event. Furthermore, the remaining $p-q$ diagonals are inserted each at any $n-p-r+1$ vertices available along the vertical runs, including the last vertex, i.e. the end point of the r^{th}, last arrival run.

Then, for $r \ge 1$ and $m > k_0$,

$$LP_3\big(k_0,m,n;p,q;r,\underset{\sim}{L}^*\big) = \sum_{(R_7^2,R_8^2)} \begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix} \tag{3.9}$$

where

$$R_7^2 = \Big\{\big(i_1,i_2,\ldots,i_q\big) : 1 \le i_1 < i_2 < \ldots < i_{q-1} < i_q, i_q = r\Big\},$$
$$R_8^2 = \Big\{p_i = \big(p_{i_1},p_{i_2},\ldots,p_{i_q}\big) : \triangle \le p_{i_s} \le l_{i_s},\ s = 1,2,\ldots,q-1,\ \text{and}\ p_{i_q} = l_{i_q}\Big\},\quad \triangle = \begin{cases} 0, & \text{if } i_s > 1, \\ k_0, & \text{if } i_s = 1. \end{cases}$$

Proof. Following generic steps used for proofing Theorem (3.4.1), hence for a lattice path satisfying SP2 and ending with a diagonal preceded by an arrival under phase 1 of service, only one diagonal out of q diagonals can be inserted at the last AR. This insertion must take place at the right end vertex of last

AR. Hence $p_{i_q} = l_{i_q}$. The proof then can be completed like in the proof for Theorem 3.4.2. □

Lemma 3.4.5. Let $LP_3(k_0, m, n; p)^*$ be the number of LP_s from $(k_0, 0)$ to (m, n), $m > n$, remaining below the line $Y = X$, each comprising of $m - p$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - p$ vertical steps and p diagonals, ending with a diagonal preceded by a departure, then summing (3.9) over r, q and L, we find

$$LP_3(k_0, m, n; p)^* = \sum_{(R_4, R_5, R_6^*)} LP_3\big(k_0, m, n; p, q; r, \underset{\sim}{L}\big)^*. \tag{3.10}$$

Lemma 3.4.6. For the case $r \geq 1$ and $p = 0$, we get

$$\begin{aligned} LP_1(k_0, m, n; 0, 0) &= LP_2(k_0, m, n; 0, 0) = LP_1^*(k_0, m, n; 0, 0) = LP_2^*(k_0, m, n; 0, 0) \\ &= LP_3^*(k_0, m, n; 0, 0) = \binom{m+n-k_0}{n} - \binom{m+n-k_0}{m}. \end{aligned} \tag{3.11}$$

3.4.2 Probability density function for pure incomplete busy period (PIBP)

Theorem 3.4.6. (Probability density function of PIBP system)
Let $f_{k_0}(t)$ denote the probability density function that the system M/C_2/1 starting initially with k_0 customers is still in continuous service at time t (PIBP). Let $f_{k_0}^1(t)$, $f_{k_0}^2(t)$, $f_{k_0}^3(t)$, $f_{k_0}^4(t)$, $f_{k_0}^5(t)$, and $f_{k_0}^6(t)$ denote the pdf for various scenarios as described in Table 3.3.
Then we have

$$f_{k_0}(t) = \sum_{i=1}^{6} f_{k_0}^i(t) \tag{3.12}$$

where

$$\begin{aligned} f_{k_0}^1(t) =& e^{-(\lambda+\mu_1)t} \sum_{m=k_0+1}^{\infty} \sum_{n=0}^{m-1} \left\{ \binom{m+n-k_0}{n} - \binom{m+n-k_0}{m} \right\} \lambda^{m-k_0} (\beta\mu_1)^n \\ &\times \frac{t^{m+n-k_0-1}}{\Gamma(m+n-k_0)}, t > 0, \end{aligned} \tag{3.13a}$$

Table 3.3: The pdf of the system $M/C_2/1$ starts initially with k_0 customers is still in continuous service at time t for various cases

Pdf	**Case**
$f^1_{k_0}(t)$	There is no customer who enters into phase 2 of service
$f^2_{k_0}(t)$	The corresponding LP satisfies SP1 with last departure event
$f^3_{k_0}(t)$	The corresponding LP satisfies SP2 with last arrival event under phase 1 of service
$f^4_{k_0}(t)$	The corresponding LP satisfies SP2 with last arrival event under phase 2 of service
$f^5_{k_0}(t)$	The corresponding LP satisfies SP1 with last diagonal event preceded by a departure
$f^6_{k_0}(t)$	The corresponding LP satisfies SP2 with last diagonal event preceded by an arrival

$$\begin{aligned} f^2_{k_0}(t) =& e^{-(\lambda+\mu_1)t} \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8)} \begin{pmatrix} n-p-r \\ p-q \end{pmatrix} \alpha_1^p \beta^{n-2p} \lambda^{m-p-k_0} \\ & \times \mu_1^{n-p} \mu_2^p \sum_{x=0}^{\infty} \frac{(\mu_2-\mu_1)^x}{x!} \frac{t^{m+n-p-k_0+x-1}}{\Gamma(m+n-p-k_0+x)} \\ & \times \frac{\Gamma(m+n-2p-k_0+x-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}))}{\Gamma(m+n-2p-k_0-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}))}, t>0, \end{aligned} \tag{3.13b}$$

$$\begin{aligned} f^3_{k_0}(t) =& e^{-(\lambda+\mu_1)t} \sum_{(R_1^2,R_2^2,R_3,R_4,R_5,R_6^*,R_7^1,R_8)} \begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix} \alpha_1^p \beta^{n-2p} \lambda^{m-p-k_0} \\ & \times \mu_1^{n-p} \mu_2^p \sum_{x=0}^{\infty} \frac{(\mu_2-\mu_1)^x}{x!} \frac{t^{m+n-p-k_0+x-1}}{\Gamma(m+n-p-k_0+x)} \\ & \times \frac{\Gamma(m+n-2p-k_0+x+1-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}))}{\Gamma(m+n-2p-k_0+1-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}))}, t>0, \end{aligned} \tag{3.13c}$$

$$f^4_{k_0}(t) = e^{-(\lambda+\mu_1)t} \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6^*,R_7^2,R_8^1)} \begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix} \alpha_1^p \beta^{n-2p+1} \lambda^{m-p-k_0}$$
$$\times \mu_1^{n-p+1} \mu_2^{p-1} \sum_{x=0}^{\infty} \frac{(\mu_2-\mu_1)^x}{x!} \frac{t^{m+n-p-k_0+x-1}}{\Gamma(m+n-p-k_0+x)}$$
$$\times \frac{\Gamma(m+n-2p-k_0+x+1-\sum_{s=1}^{q-1}(l_{i_s}-p_{i_s}))}{\Gamma(m+n-2p-k_0+1-\sum_{s=1}^{q-1}(l_{i_s}-p_{i_s}))}, t>0, \tag{3.13d}$$

$$f^5_{k_0}(t) = e^{-(\lambda+\mu_1)t} \sum_{(R_1^3,R_2^4,R_3,R_4,R_5,R_6,R_7,R_8)} \begin{pmatrix} n-p-r \\ p-q-1 \end{pmatrix} \alpha_1^p \beta^{n-2p+1} \lambda^{m-p-k_0}$$
$$\times \mu_1^{n-p+1} \mu_2^{p-1} \sum_{x=0}^{\infty} \frac{(\mu_2-\mu_1)^x}{x!} \frac{t^{m+n-p-k_0+x-1}}{\Gamma(m+n-p-k_0+x)}$$
$$\times \frac{\Gamma(m+n-2p-k_0+x+1-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}))}{\Gamma(m+n-2p-k_0+1-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}))}, t>0, \tag{3.13e}$$

$$f^6_{k_0}(t) = e^{-(\lambda+\mu_1)t} \sum_{(R_1^2,R_2^2,R_3,R_4,R_5,R_6^*,R_7^2,R_8^2)} \begin{pmatrix} n-p-r+1 \\ p-q \end{pmatrix} \alpha_1^p \beta^{n-2p} \lambda^{m-p-k_0}$$
$$\times \mu_1^{n-p} \mu_2^{p} \sum_{x=0}^{\infty} \frac{(\mu_2-\mu_1)^x}{x!} \frac{t^{m+n-p-k_0+x-1}}{\Gamma(m+n-p-k_0+x)}$$
$$\times \frac{\Gamma(m+n-2p-k_0+x+1-\sum_{s=1}^{q-1}(l_{i_s}-p_{i_s}))}{\Gamma(m+n-2p-k_0+1-\sum_{s=1}^{q-1}(l_{i_s}-p_{i_s}))}, t>0, \tag{3.13f}$$

where

$R_1^1 = \{m : \max(3, k_0+1) \le m < \infty\}$,

$R_1^2 = \{m : \max(4, k_0+1) \le m < \infty\}$,

$R_1^3 = \{m : \max(5, k_0+1) \le m < \infty\}$,

$R_2^1 = \{n : 1 \le n \le m-1\}$,

$R_2^2 = \{n : 2 \le n \le m-1\}$,

$R_2^3 = \{n : 3 \le n \le m-1\}$,

$R_2^4 = \{n : 4 \le n \le m-1\}$,

$R_3 = \{p : 1 \le p \le min(\lfloor \frac{n}{2} \rfloor, m-k_0)\}$, $\lfloor x \rfloor$ denotes the largest integer contained in x.

Proof. **First term** $(f^1_{k_0}(t), 3.13a)$. First term of (3.12) corresponds to the case when no customer enters phase 2 service. The number of arrivals is $m - k_0$, and the number of departures is n. Therefore, the total number of transitions during PIBP is $m + n - k_0$. Call this as N_0.

Let $T_0 = 0$ and $T_1, T_2, \ldots$ be the sequence of times at which the transitions occur. Let at time T_0, Poisson process starts with rate $(\lambda + \mu_1)$. The probability of an arrival occurring is $\left(\frac{\lambda}{\lambda+\mu_1} \right)$ and the probability of a departure occurring is $\left(\frac{\beta\mu_1}{\lambda+\mu_1} \right)$.

Let t be the total time spend in the system, the probability density function of t is N_0-*Erlang* with parameter $(\lambda + \mu_1)$ given by

$$f_{N_0}(t) = \frac{e^{-(\lambda+\mu_1)t}(\lambda+\mu_1)^{N_0} t^{N_0-1}}{\Gamma(N_0)}. \tag{3.14}$$

The density function for PIBP becomes

$$f^1_{k_0}(t) = \sum_{m=k_0+1}^{\infty} \sum_{n=0}^{m-1} \left\{ \binom{m+n-k_0}{n} - \binom{m+n-k_0}{m} \right\} \left(\frac{\lambda}{(\lambda+\mu_1)} \right)^{m-k_0} \left(\frac{\beta\mu_1}{(\lambda+\mu_1)} \right)^{n} \frac{e^{-(\lambda+\mu_1)t}(\lambda+\mu_1)^{m+n-k_0} t^{m+n-k_0-1}}{\Gamma(m+n-k_0)} \tag{3.15}$$

where $\binom{m+n-k_0}{n} - \binom{m+n-k_0}{m}$ is the number of lattice paths starting from $(k_0, 0)$ to (m, n) always remaining below the line $Y = X$ (Sen and Jain, 1993).

After simplification, we find

$$f^1_{k_0}(t) = \sum_{m=k_0+1}^{\infty} \sum_{n=0}^{m-1} \left\{ \binom{m+n-k_0}{n} - \binom{m+n-k_0}{m} \right\} \lambda^{m-k_0} (\beta\mu_1)^n t^{m+n-k_0-1} \frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-k_0)}$$

which is (3.13a).

Second term $(f^2_{k_0}(t), 3.13b)$. To prove the second term of (3.12), let t_1 be the total time spent in phase 1 of service and $t - t_1$ the total time spent in phase 2 of service out of the total time t spent in the system. We consider here that the last event is departure after a customer completes phase 1 of service.

For fixed $\underset{\sim}{L}$, total number of transitions during the PIBP of length t is given by $m+n-p-k_0$ as described below:

Number of arrivals: $m-p-k_0$.

Number of departures after phase 1 : $n-2p$.

Number entries into phase 2: p.

Number of departures after phase 2: p.

The total number of transitions in t_1, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as well as entries into phase 2 of service as explained below:

Number of arrivals: $m-p-k_0-\sum_{s=1}^{q}(l_{i_s}-p_{i_s})$ since $\sum_{s=1}^{q}(l_{i_s}-p_{i_s})$ is the number of arrivals while customers are in phase 2 of service.

Number of entries into phase 2: p.

Number of departures: $n-2p$.

Hence we obtain N_1, total number of transitions during t_1 as below.

$$N_1 = m+n-2p-k_0-\sum_{s=1}^{q}(l_{i_s}-p_{i_s}). \tag{3.16}$$

The probability density function of t_1 is N_1-*Erlang* with parameter $(\lambda+\mu_1)$ given by

$$f_{N_1}(t_1) = \frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}t_1^{N_1-1}}{\Gamma(N_1)}. \tag{3.17}$$

The total number of transitions in $t_2 = t-t_1$, the time spent in phase 2 of service, consists of arrivals and departures while customers are in phase 2 of service as explained below:

Number of arrivals: $\sum_{s}^{q}\big(l_{i_s}-p_{i_s}\big)$.

Number of departures: p.

Therefore N_2, total number of transitions during $t_2 = t-t_1$ will be

$$N_2 = \sum_{s=1}^{q}\big(l_{i_s}-p_{i_s}\big)+p. \tag{3.18}$$

The probability density function of t_2 is N_2-*Erlang* with parameter $(\lambda+\mu_2)$ given by

$$f_{N_2}(t_2) = \frac{e^{-(\lambda+\mu_2)t_2}(\lambda+\mu_2)^{N_2}t_2^{N_2-1}}{\Gamma(N_2)}. \tag{3.19}$$

For the second case, for combining the duration of phase 1 and phase 2 of the service, we get

$$
\begin{aligned}
f^2_{k_0}(t) = & \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8)} \begin{pmatrix} n-p-r \\ p-q \end{pmatrix} \int_0^t f_{N_1}(t_1)\left(\frac{\lambda}{\lambda+\mu_1}\right)^{N_1-n+p} \\
& \left(\frac{\alpha_1\mu_1}{\lambda+\mu_1}\right)^p \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n-2p} f_{N_2}(t-t_1)\left(\frac{\lambda}{\lambda+\mu_2}\right)^{N_2-p} \left(\frac{\mu_2}{\lambda+\mu_2}\right)^p dt_1 \\
= & \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8)} \begin{pmatrix} n-p-r \\ p-q \end{pmatrix} \alpha_1^p \beta^{n-2p}\lambda^{N_1+N_2-n}\mu_1^{n-p}\mu_2^p \\
& \frac{1}{\Gamma(N_1)}\frac{1}{\Gamma(N_2)}\int_0^t e^{-(\lambda+\mu_1)t_1}e^{-(\lambda+\mu_2)(t-t_1)}t_1^{N_1-1}(t-t_1)^{N_2-1}dt_1 \\
= & \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8)} \begin{pmatrix} n-p-r \\ p-q \end{pmatrix} \alpha_1^p \beta^{n-2p}\lambda^{N_1+N_2-n}\mu_1^{n-p}\mu_2^p \\
& \frac{1}{\Gamma(N_1)}\frac{1}{\Gamma(N_2)}e^{-(\lambda+\mu_2)t}\int_0^t e^{(\mu_2-\mu_1)t_1}t_1^{N_1-1}(t-t_1)^{N_2-1}dt_1.
\end{aligned}
\tag{3.20}
$$

The integral part can be simplified as below.

$$
\begin{aligned}
\int_0^t e^{(\mu_2-\mu_1)t_1}t_1^{N_1-1}(t-t_1)^{N_2-1}dt_1 &= \int_0^t \sum_{x=0}^{\infty}\frac{\big((\mu_2-\mu_1)t_1\big)^x}{x!}t_1^{N_1-1}(t-t_1)^{N_2-1}dt_1 \\
&= \int_0^t \sum_{x=0}^{\infty}\frac{\big(\mu_2-\mu_1\big)^x}{x!}t_1^{N_1+x-1}(t-t_1)^{N_2-1}dt_1
\end{aligned}
$$

by substituting $v=\frac{t_1}{t}$ then we get

$$
\begin{aligned}
\int_0^t e^{(\mu_2-\mu_1)t_1}t_1^{N_1-1}(t-t_1)^{N_2-1}dt_1 &= \sum_{x=0}^{\infty}\frac{\big(\mu_2-\mu_1\big)^x}{x!}t^{N_1+N_2+x-1}\int_0^1 v^{N_1+x-1}(1-v)^{N_2-1}dv \\
&= \sum_{x=0}^{\infty}\frac{\big(\mu_2-\mu_1\big)^x}{x!}t^{N_1+N_2+x-1}\beta\big(N_1+x,N_2\big)
\end{aligned}
$$

where

$$\beta(x,y)=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}.$$

Finally, substituting the above formula in (3.20) we get $f^2_{k_0}(t)$. □

Third term ($f^3_{k_0}(t), 3.13c$). This term corresponds to the case when last event is an arrival in phase 1 of service. The proof can be done by following to the proof of First term. Hence, the number of entries into phase 2 is p and the number of departures after phase 2 is p. Again, we decompose the total time t into t_1 and $t_2 = (t - t_1)$ as time spend in phase 1 and phase 2, respectively. In this case the number of transition during t_1 and t_2 are

$$N_1 = m + n - 2p - k_0 - \sum_{s=1}^{q}(l_{i_s} - p_{i_s})$$

and

$$N_2 = \sum_{s=1}^{q}\big(l_{i_s} - p_{i_s}\big) + p,$$

respectively. Finally, combining the duration of phase 1 and phase 2 service as in equation (3.20) and using Theorem 3.4.2, we get (3.13c).

Fourth term $(f^4_{k_0}(t), 3.13d)$. This term corresponds to the case when last event is an arrival in phase 2 of service. The proof can be done by following to the proof of First term. Hence, the number of entries into phase 2 is p and the number of departures after phase 2 is $p - 1$. Again, we decompose the total time t into t_1 and $t_2 = (t - t_1)$ as time spend in phase 1 and phase 2, respectively. In this case the number of transition during t_1 and t_2 are

$$N_5 = m + n - 2p - k_0 - \sum_{s=1}^{q-1}(l_{i_s} - p_{i_s}) + 1$$

and

$$N_6 = \sum_{s=1}^{q-1}\big(l_{i_s} - p_{i_s}\big) + p - 1,$$

respectively. Finally, combining the duration of phase 1 and phase 2 service as in equation (3.20) and using Theorem 3.4.3, we get (3.13d).

Fifth term $(f^5_{k_0}(t), 3.13e)$ This term corresponds to the case when last event is a diagonal preceded by a departure event. The proof can be done by following to the proof of First term. Hence, the number of entries into phase 2 is p and the number of departures after phase 2 is $p - 1$. Again, we decompose the total time t into t_1 and $t_2 = (t - t_1)$ as time spend in phase 1 and phase 2, respectively. In this case the number of transition during t_1 and t_2 are

$$N_7 = m + n - 2p - k_0 - \sum_{s=1}^{q}(l_{i_s} - p_{i_s}) + 1$$

and

$$N_8 = \sum_{s=1}^{q} \big(l_{i_s} - p_{i_s}\big) + p - 1,$$

respectively. Finally, combining the duration of phase 1 and phase 2 service as in equation (3.20) and using Theorem 3.4.4, we get (3.13e).

Sixth term $(f^6_{k_0}(t), 3.13f)$. This term corresponds to the case when last event is a diagonal preceded by an arrival event. The proof can be done by following to the proof of First term. Hence, the number of entries into phase 2 is p and the number of departures after phase 2 is $p-1$. Again, we decompose the total time t into t_1 and $t_2 = (t - t_1)$ as time spend in phase 1 and phase 2, respectively. In this case the number of transition during t_1 and t_2 are

$$N_9 = m + n - 2p - k_0 - \sum_{s=1}^{q-1} (l_{i_s} - p_{i_s}) + 1$$

and

$$N_{10} = \sum_{s=1}^{q-1} \big(l_{i_s} - p_{i_s}\big) + p - 1,$$

respectively. Finally, combining the duration of phase 1 and phase 2 service as in equation (3.20) and using Theorem 3.4.5, we get (3.13f).

3.5 Numerical Computations and Comments

An R program has been developed for numerical computation of equation (3.12). The program starts with generating all possible lattice paths using the library AlgDesign. Next, only the paths satisfying the SP1, SP2 and SP3 are filtered. These paths form the set L and, finally, equation (3.12) is computed for the selected paths.

The code for computing the density is presented in Appendix A1. On a computation process for $t = 0.05(0.01)1.5$, the computation takes < 3 CPU minutes.

The outputs of numerical computations for different parameters involved are presented in Figure 3.6 - Figure 3.10. From these graphs, for $0 < t \leq 1.5$ we notice that as t increases, the density function of PIBP system computed for a given

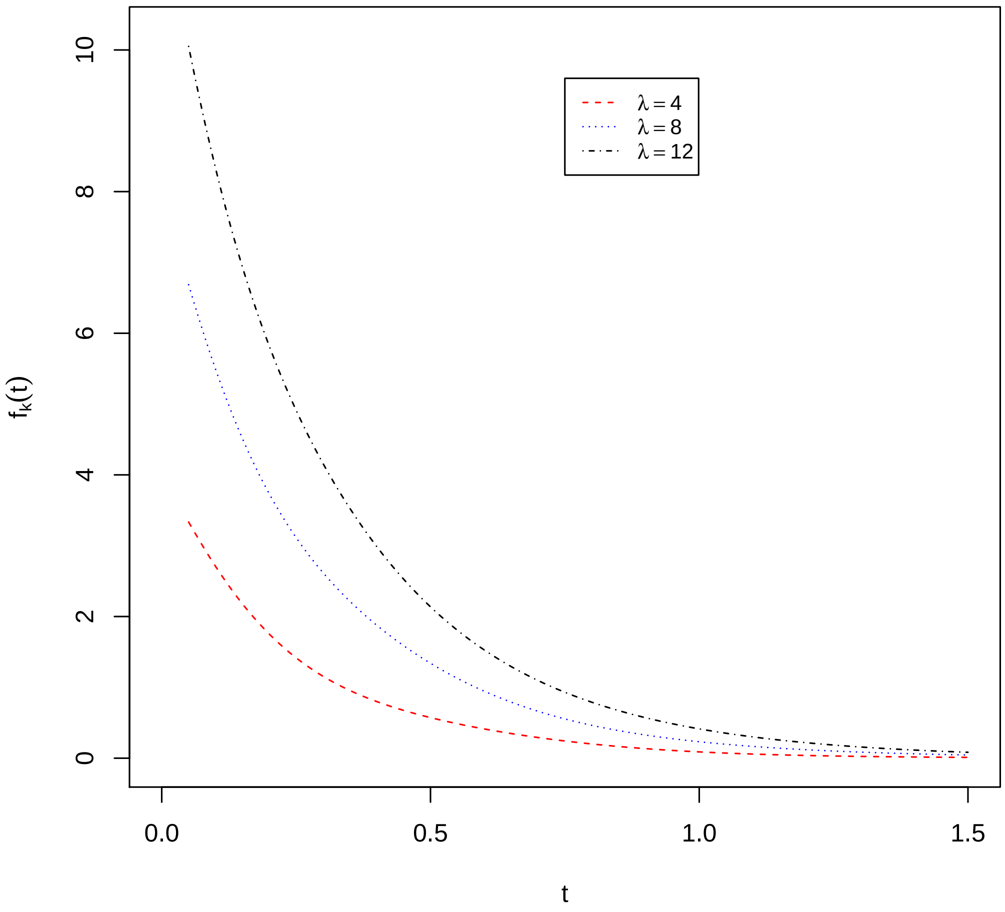


Figure 3.6: Density of the pure incomplete busy period system of $M/C_2/1$ queue, $f_{k0(t)}$, for different values of λ taking $k_0 = 1$, $\mu_1 = 8$, $\mu_2 = 4$, $\lambda = 5$, $\alpha = 0.4$, $\beta_1 = 0.6$

time t decreases, i.e. satisfies the expected pattern that at a larger unit of time we expect PIBP density function to cease, leading to zero probability at such time i.e. $t = 1.5$.

From Figure 3.6, for different values of λ we notice that as t increases, the density function of PIBP system decreases. The maximum value of density increases as the value of λ increases. The rate of decrease in the value of density for smaller value of λ is less than that for the larger value of λ.

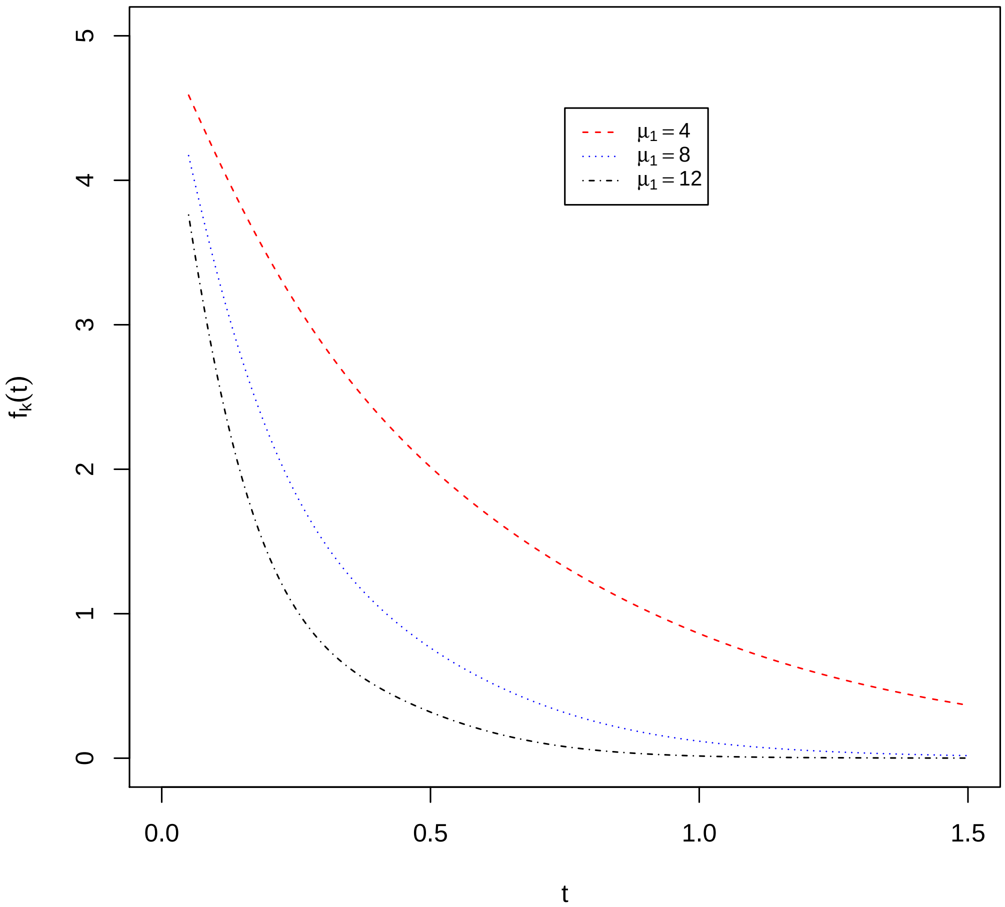


Figure 3.7: Density of the pure incomplete busy period system of $M/C_2/1$ queue, $f_{k_0(t)}$, for different values of μ_1 taking $k_0 = 1,\ \mu_2 = 4,\ \lambda = 5,\ \alpha = 0.4,\ \beta_1 = 0.6$

It can be seen from Figure 3.7 for different values of μ_1 that as t increases, the density function of PIBP system decreases. The rate of decrease in the value of density for smaller μ_1 is less than for larger value of μ_1.

For different values of μ_2 as in Figure 3.8, we notice that as t increases, the density function of PIBP system decreases. As μ_2 increases, the density also increases. But for a particular value of t, we notice that PIBP system increases with a decrease in the value of μ_2.

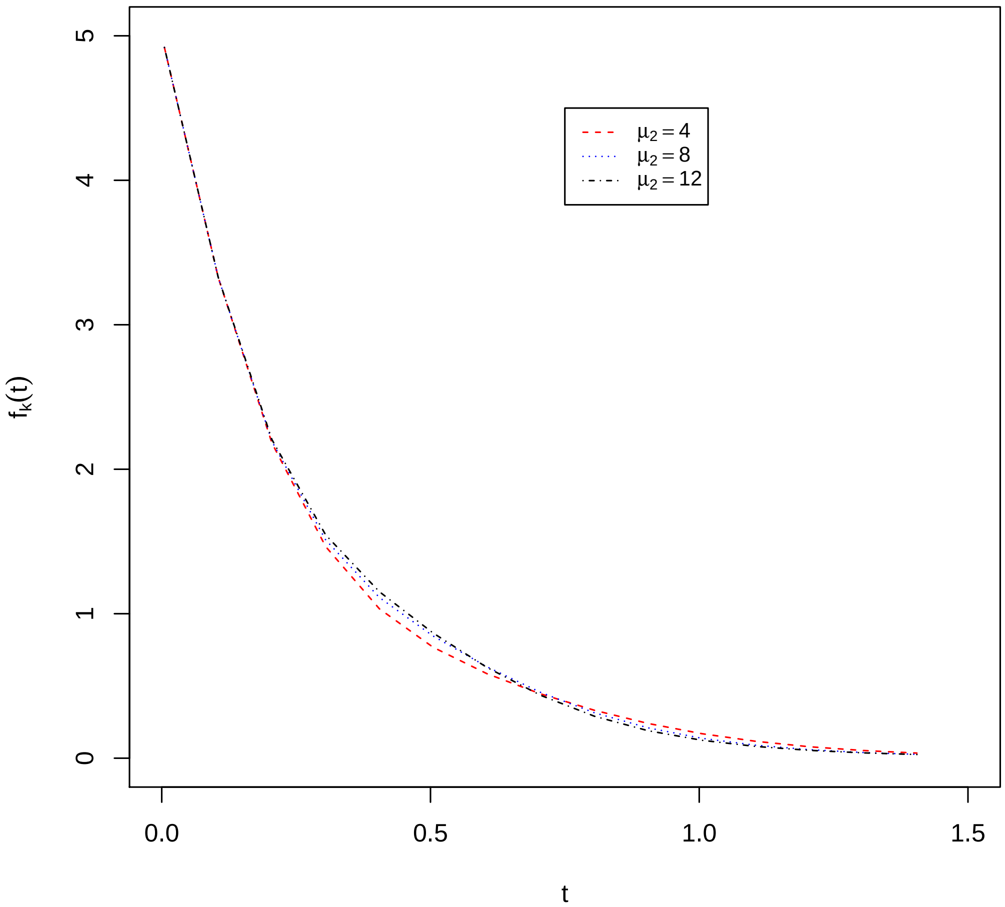


Figure 3.8: Density of the pure incomplete busy period system of $M/C_2/1$ queue, $f_{k0(t)}$, for different values of μ_2 taking $k_0 = 1,\ \mu_1 = 8,\ \lambda = 5,\ \alpha = 0.4,\ \beta_1 = 0.6$

As in Figure 3.9 for different values of α we notice that as t increases, the density function of PIBP system initially decreases. Increase in α means more and more customers enter the next phase of service. The rate of decrease in the value of density for smaller value of α is less than that for the larger value of α.

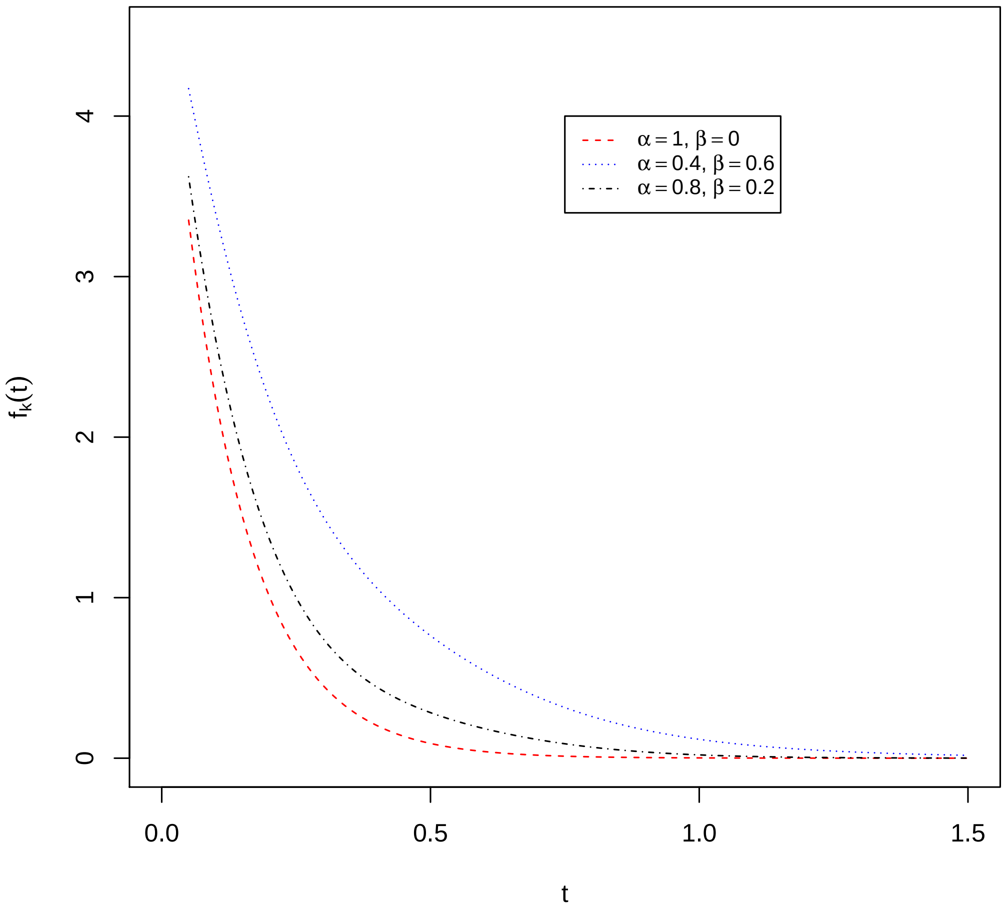


Figure 3.9: Density of the pure incomplete busy period system of $M/C_2/1$ queue, $f_{k0(t)}$, for different values of α $(\beta = 1 - \alpha$ taking $k_0 = 1,\ \mu_1 = 8,\ \mu_2 = 10,\ \lambda = 5$

For different values of k_0 we notice from Figure 3.10 that as k_0 increases, the density function of PIBP system increases. As k_0 increases, the rate of decrease in the value of density for smaller value of k_0 is greater than that for the larger value of k_0.

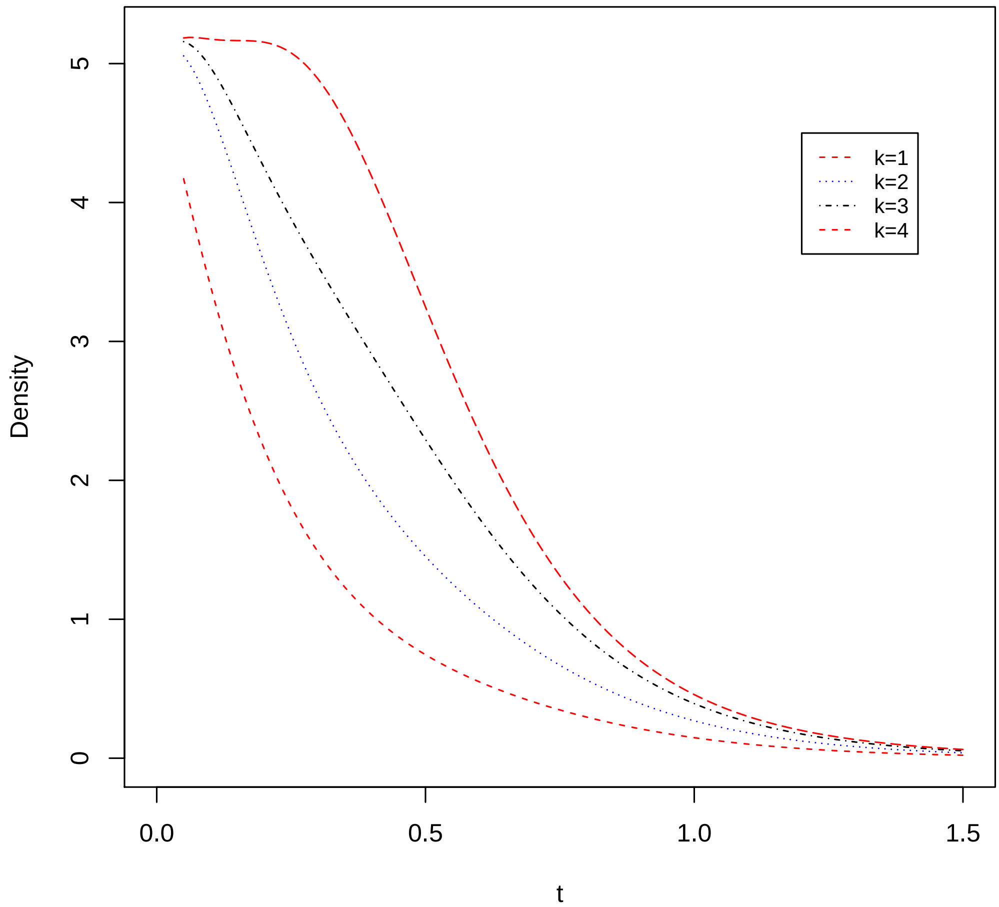


Figure 3.10: Density of the pure incomplete busy period system of $M/C_2/1$ queue, $f_{k_0(t)}$, for different values of k_0 taking $\mu_1 = 8,\ \mu_2 = 10,\ \lambda = 5,\ \alpha = 0.4,\ \beta_1 = 0.6$

3.6 Summary

In this chapter, we derived the pure incomplete busy period density function of $M/G/1$ queues, through LPC approach which entails approximating general service distribution by Coxian type-phase distribution. This includes:

(i) Constructing a complete set of structural properties for lattice path representations with last departure, last arrival and last diagonal events. This includes defining the basic notation and terminology, representing the pure incomplete busy period, counting the lattice paths, constructing the run, inserting the diagonals, and approximating the service time by Coxian 2-phase distribution.

(ii) Deriving the enumeration of the number of lattice paths corresponding to its structural properties.

(iii) Obtaining the pure incomplete busy period density function of $M/G/1$ queues by combining steps (i) and (ii).

To achieve better understanding of the behavior of the incomplete busy period density function obtained, numerical computations for parameters involved have been performed. The required algorithms and the computational procedures are developed as R programs [97].

The investigation of the influence of taking different values of parameters on the behaviour of the graphs of the density function shows that as t increases, the density function of PIBP system $f_{k_0}(t)$ computed for $t = 0.005(0.1)1.5$ decreases.

Chapter 4

M/G/1 QUEUE UNDER $(0, k)$ CONTROL POLICY

4.1 Introduction

In real life situations often queueing systems are operated under control policies on service mechanisms to optimally use the service resources. But the literature hardly presents explicit expression for the of incomplete busy period densities of a $M/G/1$ queueing process under control policies.

In this chapter, our main objective is to compute the probability density function of IBP of an $M/G/1$ queue operating under control policies through LPC approach which entails approximating general service distribution by C_2, Coxian 2-phase distribution. We focus on modeling the transient behaviour of $M/G/1$ queues under $(0, k)$ control policy, wherein the server goes on the vacation when the system becomes empty and re-opens for service immediately at the arrival of the k^{th} costumer.

The queueing system under control policy has been investigated by numerous researchers. Choudhury and Tadj [21] studied the steady state behaviour of $M/G/1$ queuing system with an additional second phase of optional service subject to breakdowns occurring randomly at any instant while serving the customers and delayed repair. Agarwal and Dshalalow [3] studied the D-control policy with multiple vacations for $M/G/1$ queuing system. Wang and Huang [102] considered the optimal control for $M/E/1$ queueing system with a removable service station, and Wang and Ke [103] studied optimal control of the $M/G/1$ models for both finite and infinite capacity. Sen and Gupta [86, 87] have investigated and derived transient solution for $M/M/1$ queueing systems with $T-$policy through LPC approach while Sen et al. [89] have considered the system under $(0, k)$ control policy for $M/M/1$ queueing model. Wazalwar and Khaparde [104] have derived busy period density function for $M/C_2/1$ queues under $(0, k)$ control policy where arrival and service process are correlated.

Heyman [44] was first to consider the economic behaviour of the $M/G/1$ queueing system under $(0, k)$ control policy. For the steady state conditions, the results presented were an exact expression for the optimal value of k, the number of customers in the system and the optimal value of the cost rate as a function of k.

Explicit form of the incomplete busy period density and other measures of the system performance are not known for $M/G/1$ queues that entail systems with general service time distribution and $(0, k)$ control policy. The major aim of this chapter is thus to extend LPC approach and obtain the expression for the density of the incomplete busy period of $M/G/1$ queueing system under $(0, k)$ policy.

The rest of the chapter is organized as follows. In the following section, we recapitulate the methodology for determining the probability density function, definition of lattice path and briefly explain its application to determine the density function of IBP of queueing system under $(0, k)$ policy. Section 4.3 presents the results on the number of lattice paths and their use to compute the transient probabilities. In section 4.4, we present the numerical computation of density for IBP of $M/G/1$ queueing system under $(0, k)$ policy. Finally the summary is presented in section 4.5.

4.2 The $M/C_2/1$ Model under $(0, k)$ Control Policy

We recapitulate the methodology described in Section (3.3) to derive the probability density function of the incomplete busy period of $M/G/1$ queueing system under $(0, k)$ policy. As in the previous chapter, see Section (3.2), $M/G/1$ model will be approximated as $M/C_2/1$. That is general service is approximated as Coxian 2-phase model.

Let $f_{k_0;k}(t)$ denote the pdf of IBP of $M/C_2/1$ model under $(0, k)$ control policy for $t \geq 0$, where k_0 is the initial number of customers in the system and k is the number of arrivals required during the vacation period for the queueing system to restart the busy period. A realization of the queueing process of $M/C_2/1$ during a IBP may be represented by a lattice path in a two dimensional plane starting at $(k_0, 0)$ and ending at (m, n). The queueing process is observed at the points of transitions over time interval $(0, t)$. At this stage, an arrival, a departure or shifting of a customer to phase 2 on the completion of phase 1 service is represented by a unit of horizontal, unit of vertical or $\sqrt{2}$ unit of diagonal steps, respectively.

In a lattice path representation of a $M/G/1$ queueing system, a vertex (x, y) has the following interpretation:

$x =$ the sum of number of arrivals, number of initial customers and number of customers shifting to phase 2 service,

$y =$ the sum of number of departures and number of customers shifting to phase 2 service.

Thus, in the lattice path we have $x \geq y$, in other words lattice path will never cross the barrier $Y = X$ in the plane, but may touch it.

Thus, an IBP during $(0, t)$ with k_0 initial customers in the system is represented by a lattice path starting from $(k_0, 0)$ to (m, n), $m > n$, always remaining below the barrier $Y = X$. Such a lattice path may touch the barrier $Y = X$, say s times and satisfies the following properties:

(i) After each of the s touches with the barrier, the path has at least k horizontal steps preceding the first vertical step (or no vertical step).

(ii) After the last (s^{th}) touch of the barrier $Y = X$, the path ends with $k^* < k$ horizontal steps up to (m, n). When $k^* \geq k$, the, system is busy at the end On the other hand the system ends in vacation .

Hence, in the lattice path under $(0, k)$ control policy for s touches, we have $s-1$ closed cycles i.e $s-1$ complete busy period (of ARs $\underset{\sim}{u}$) and 1 open cycle i.e 1 pure incomplete busy period of (AR $\underset{\sim}{u}^*$).

The IBP of a $M/G/1$ queueing system under $(0, k)$ control policy with $k_0 = 2$ and $k = 2$ is represented by lattice path, as illustrated in Figure 4.1. In the graph of the lattice path we represent an arrival or departure occurring during the progress of phase 1 (phase 2) service by solid (dotted) horizontal or solid (dotted) vertical line. Furthermore, a shift to phase 2 service from phase 1 is denoted by solid diagonal line in the LP. Note that in Figure 4.1 server is active (in serving mode) at the end point of the path. The number of departures is same as the number of arrivals at some points, but the lattice path never reaches the point $(27, 27)$.

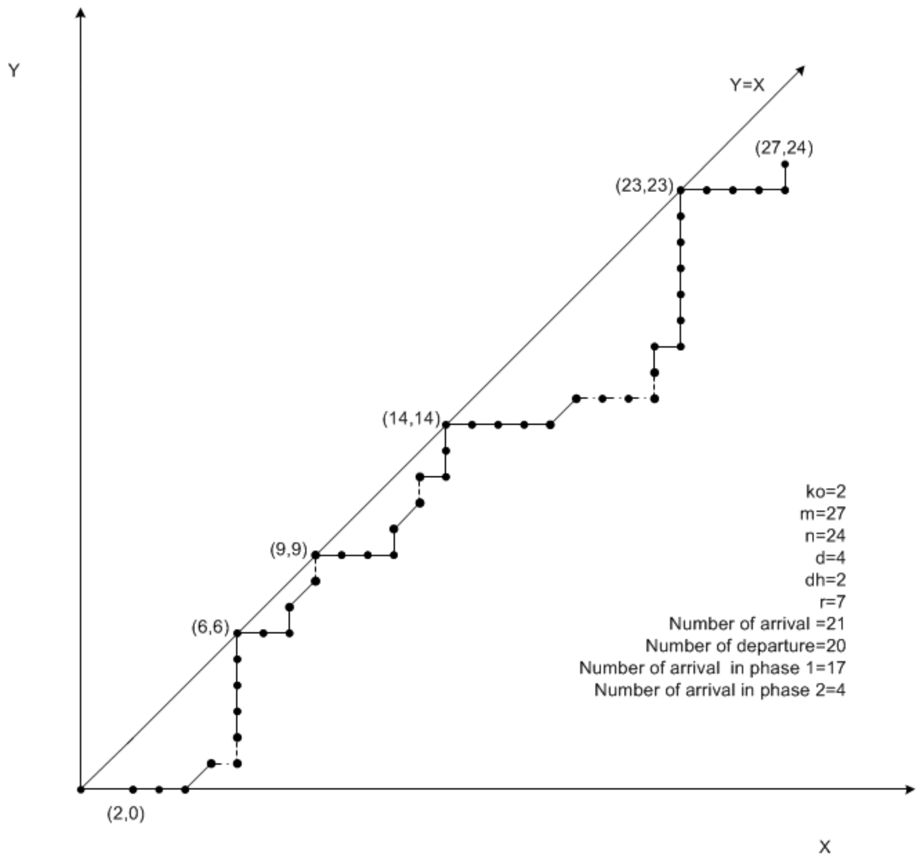


Figure 4.1: $M/C_2/1$ model. An example of lattice path representation under $(0, k)$ control policy

Such a lattice path (corresponding to a IBP during $(0, t)$ with k_0 initial customers in the system) will either end with a vertical step (departure), a horizontal step (arrival) or a diagonal step (shift to phase 2 service).

The set of all lattice paths satisfying the properties of IBP period over time $(0, t)$ can be partitioned into three cases as follows:

(i) The set of lattice paths ending with solid (or dashed) vertical line corresponding to a departure after phase 1 (or phase 2) service.

(ii) The set of lattice paths ending with solid (or dashed) horizontal line corresponding to an arrival during phase 1 (or phase 2) service.

(iii) The set of lattice paths ending with a solid diagonal line corresponding to a customer shifting to phase 2 of service. Note that such a solid diagonal line can occur immediately after a solid vertical line (a departure after phase 1 service), dotted vertical line (a departure after phase 2 service) or solid horizontal line (an arrival during the operation of phase 1 service).

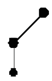
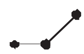
Table 4.1 illustrates the decomposition of lattice paths into disjoint groups as explained above in (i)-(iii). The last column provides the reference to the theorem where the counting results on the corresponding lattice paths are established. Each of these steps is illustrated in details over the remaining sections.

Next, the pdf $f_{k_0;k}(t)$ is estimated as the proportion of number of lattice paths satisfying the properties of IBP over $(0, t)$. This is accomplished by counting the number of lattice paths satisfying the properties of IBP in Theorems 4.4.1, 4.4.2, 4.4.3, 4.4.4, 4.4.5 and 4.4.6 corresponding to the ending structure of the lattice paths. Finally, the probabilities corresponding to such paths are computed using transition probabilities corresponding to $M/C_2/1$ model to arrive at $f_{k_0;k}(t)$.

4.2.1 Transitions

The approach to investigate the transient solution is similar to the method described in Section 3.3.1. The staying time in each state is an exponential random

Table 4.1: Structural properties of LP's ending event and its related theorems

System vacation/busy	**Description of ending event of LP**	**Represen-tation**	**Theorem Reference**
Vacation period	- LP with last arrival event during vacation period		4.3.2
	- LP with last departure event after phase 1 of service		4.3.1
	- LP with last diagonal event preceded by departure after phase 1 of service		4.3.5
Busy period	- LP with last diagonal event preceded by departure after phase 2 of service		4.3.5
	- LP with last arrival event during phase 1 of service		4.3.3
	- Entry into phase 2 of service from phase 1		4.3.6
	- LP with last departure event during phase 2 of service		4.3.1
	- LP with last arrival event during phase 2 of service		4.3.4

variable as given below:

$$P\{T_{n+1} - T_n > t | X_n = i\} = \begin{cases} e^{-(\lambda+\mu_u)t}, & \text{if a customer is undergoing phase } u \\ & \text{of service, } u\text{=1,2} \\ e^{-\lambda t}, & \text{if the system is under vacation} \end{cases}$$

4.2.2 Counting of lattice paths

In this section, we present the process of counting the lattice paths for a given scenario. First, we need to get a skeleton path. For the purpose of counting of LPs, we first transform lattice path (as in Figure 4.1 to a skeleton lattice path (SLP) by removing all diagonals. A skeleton path corresponding to Figure 4.1 is illustrated in Figure 4.2. Now, it can be seen from Figure 4.2 that this skeleton LP only consists of horizontal and vertical runs which represent arrivals in phase 1, vacation, and departures after phase 1, respectively. As we have already removed all diagonals from Figure 4.1, when we count the possible number of LPs, we need to consider all possibilities of inserting the diagonals into horizontal and/or vertical runs.

For the purpose of counting the lattice paths, we again use the notion of arrival run (AR) and departure run (DR) as defined on page 43. Also we note that the conditions of inserting diagonals follow as described in Chapter 3 page 44. Further, we note that following additional restriction on the diagonals is required due to $(0, k)$ control policy. No diagonal can be inserted in first k horizontal steps after a touch with $Y = X$.

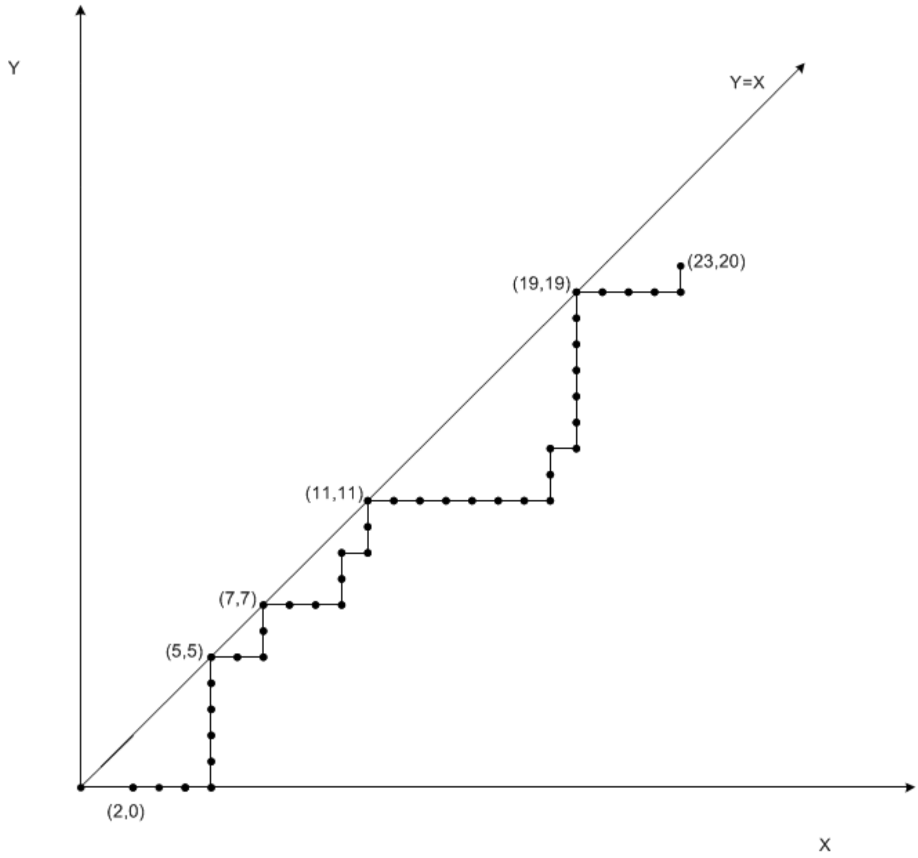


Figure 4.2: Skeleton path of an example of lattice path representation under $(0, k)$ control policy

4.2.3 Notation and terminology

Table 4.2 presents the basic notation and terminology that we use in this chapter. When possible, we retain same notation as in Chapter 3.

Table 4.2: The notation and terminology

Notation	**Description**
k_0	Initial number of customers at the start of busy period.
k	Control policy, k arrivals required during the vacation period for the queueing system to restart the busy period.
m	Total number of arrivals and customers shifting to phase 2 service.
n	Total number of departures and customers shifting to phase 2 service.
s	The number of times a LP from $(k_0, 0)$ to (m, n), $m > n$ touches the barrier line $Y = X$, $(s \geq 1)$.
r	Number of AR $(r \geq 2)$.
d	Total number of diagonals inserted in ARs and/or departure. runs $(d \geq 1)$ from $(k_0, 0)$ to (m, n).
d_h	Total number of diagonals inserted in AR $(d_h \geq 0)$ from $(k_0, 0)$ to (m, n).
$d - d_h$	Number of diagonals inserted in DR from $(k_0, 0)$ to (m, n).
l_i	Length of the ith AR $i = 1, 2, \ldots, r$.
L_i	Length of the ith DR $i = 1, 2, \ldots, r$.
$\underset{\sim}{L}$	$(l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_r)$.
$\underset{\sim}{L}^*$	$(l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_{r-1})$.
$\underset{\sim}{i}$	$\big(i_1, i_2, \ldots, i_{d_h}\big)$, d_h indices of the AR in each of which a diagonal is inserted.
$\underset{\sim}{u}$	$(u_0, u_1, \ldots, u_s)$ the index number of AR touching the barrier $Y = X$, $u_0 = 1$.
$\underset{\sim}{u}^*$	$(u_s, u_s + 1, \ldots, r)$ the index number of AR after s^{th} touch.
$l_{\underset{\sim}{i}}$	Lengths of horizontal runs $\underset{\sim}{i}$.
$l_{\underset{\sim}{u}}$	$(l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s})$, lengths of AR $\underset{\sim}{u}$.
$l_{\underset{\sim}{u}^*}$	$(l_{u_s}, l_{u_s+1}, \ldots, l_r)$, lengths of AR $\underset{\sim}{u}^*$.
$p_{\underset{\sim}{i}}$	$(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}})$, distances from extreme left end points where diagonals are inserted in horizontal runs $\underset{\sim}{i}$ including vertices at both ends of the runs).
$CC(\underset{\sim}{u})$	$s - 1$ closed cycles, complete busy period of AR $\underset{\sim}{u}$.
$OC(\underset{\sim}{u}^*)$	1 opened cycle, pure incomplete busy period of AR $\underset{\sim}{u}^*$.

To illustrate these notations, we refer to Figure 4.1. Here $k_0 = 2, k = 2, s = 4, r = 7, d = 4, d_h = 2, l_i : (5,2,3,1,7,1,4), L_i : (5,2,2,2,2,6,1), \underset{\sim}{i} : (1,5), \underset{\sim}{u} : (1,2,3,5,7), \underset{\sim}{u}^* : (7), l_{\underset{\sim}{u}} : (5,2,3,7,4), l_{\underset{\sim}{u}^*} : (7), p_{\underset{\sim}{i}} : (4,4)$. Hence, $CC(\underset{\sim}{u})$: 3 closed cycles of AR $\underset{\sim}{u}$ are

(1). $\sum_{j=1}^{1} l_j = \sum_{j=1}^{1} L_1$ where $\sum_{j=1}^{1} l_j = l_1 = 5, \sum_{j=1}^{1} L_1 = L_1 = 5.$

(2). $\sum_{j=2}^{2} l_j = \sum_{j=2}^{2} L_1$ where $\sum_{j=2}^{2} l_j = l_1 = 2 = \sum_{j=2}^{2} L_1 = L_1 = 2.$

(3). $\sum_{j=3}^{4} l_j = \sum_{j=3}^{4} L_j$ where $\sum_{j=3}^{4} l_j = l_3 + l_4 = 3+1 = 4, \sum_{j=3}^{4} L_j = L_3 + L_4 = 2+2 = 4.$

Here, $\underset{\sim}{u}^* = 7$, therefore $OC(\underset{\sim}{u}^*)$ is $l_7 = 4 > L_7 = 1$.

4.3 Results on Lattice Path Counting

The following result of lattice paths have been established which will be used in the next section to obtain the expression for the transient analysis of the system still active under $(0, k)$ policy.

4.3.1 Structural properties of lattice paths

In this section, we discuss the special feature of lattice paths corresponding to the scenario described in Table 4.1 and use them to count the number of lattice paths.

4.3.1.1 Structural properties of a LP ending with a last departure event (SPLDE)

Given non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1), l_1, l_2 \ldots, l_r$; $L_1, L_2, \ldots, L_r$, consider a lattice path under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches $Y = X$ s times but never crosses the line $Y = X$. Such a LP satisfies the following structural properties:

(a) $m - d$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq \text{Max}(k_0, L_1), l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - d$,

(b) $n - d$ vertical steps form r runs of lengths $L_1, L_2, \ldots, L_r$, respectively, satisfying $L_1, L_2, \ldots, L_r > 0$ and $\sum_{i=1}^{r} L_j = n - d$,

(c) $m - d$ horizontal steps and $n - d$ vertical steps form s closed cycles. Let $(1, u_1, u_2, \ldots, u_s)$ be the indices of the first AR in each cycle (with left ends on $Y = X$). Furthermore, $(l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s})$ are the lengths of the corresponding AR, where $l_i \geq k$, for $i = 1, u_1, u_2, \ldots, u_s$ respectively, satisfying $\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j$, $v = 0, 1, \ldots, s - 1$,

(d) The cycle at the end which is an open cycle comprising of $r - u_s + 1$ AR of lengths $l_{u_s}, l_{u_s+1}, \ldots, l_r$ respectively, satisfying $l_{u_s} \geq \text{Max}\big(k, L_{u_s}\big), \sum_{i=u_s}^{r} l_i > \sum_{i=u_s}^{r} L_i$,

(e) d_h diagonals are inserted each in any d_h out of r ARs, including the vertices at both ends of the runs except first k_0 vertices of first AR and first k vertices of u_i^{th} AR, $i = 1, 2, \ldots, s$.

The conditions (a)-(e) above would be referred to as structural properties of a LP ending with a last departure event (SPLDE) of IBP of the system under $(0, k)$ policy.

Theorem 4.3.1. (The number of LPs satisfying SPLDE)
For non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$, $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$, let $LP_1\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}\big)$ denote the number of LP$_s$ satisfying SPLDE. Furthermore, the remaining $d - d_h$ diagonals are inserted each at any $n-d-r$ vertices available along the vertical runs (excluding the last vertex i.e. the end point of the last vertical run).

Then, for $k \geq 1,\ m \geq k_0 + k,\ n \geq k_0 + 1$ and $r \geq 1$,

$$LP_1\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}\big) = \sum_{(R_8, R_9)} \binom{n - d - r}{d - d_h}, \tag{4.1}$$

where

$$R_8 = \Big\{ \big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r \Big\},$$

$$R_9 = \left\{ p_i = \left(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\right) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h \right\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s). \end{cases}$$

Proof. Before proofing the above theorem, let us consider generic steps taken to prove the theorem. These are

(i) to consider a LP from $(0, 0)$ to (m, n),

(ii) to delete all the diagonal steps,

(iii) to concatenate the steps to form a skeleton path consisting of AR and DR satisfying the properties of respective structural properties,

(iv) to enumerate the number of LP by inserting all possible diagonals.

Let us consider a LP from $(0, 0)$ to $(m, n), m > n$. Delete all the d diagonal steps, and concatenate the steps to form a skeleton path consisting of AR and DR as described by vector $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$ and satisfying the SPLDE above. We now focus on this unique skeleton path and enumerate the number of LPs by inserting diagonals as described in condition (e). Let d_h diagonals be inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, respectively with lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d - d_h$ diagonals will be inserted into any $d - d_h$ vertices out of $n - d - r$ internal vertices available in the DR. This can be accomplished in $\begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix}$ ways. Now summing $\begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ defined by R_8 and all insertion points $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ defined by R_9, we get (4.1). □

Lemma 4.3.1. Let $LP_1\big(k_0, k, m, n; d, s\big)$ be the number of LP$_s$ under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches $Y = X$ s times but never crosses the line $Y = X$ and ends with a last departure event. Then, summing (4.1) over r, d_h, and L we find

$$LP_1(k_0, k, m, n; d, s) = \sum_{(R_5, R_6, R_7)} LP_1\Big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}\Big), \qquad (4.2)$$

where

$$R_5 = \{r : 2 \leq r \leq \max(n - d - k_0, 2)\},$$
$$R_6 = \{d_h : \max(0, 2d - n + r) \leq d_h \leq \min(r, d)\},$$
$$R_7 = \Big\{\underset{\sim}{L} : l_1 \geq \text{Max}(k_0, L_1 + 1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k, v = 0, 1, \ldots, s-1, \sum_{i=u_s}^{r} l_i > \sum_{i=u_s}^{r} L_i, \sum_{i=1}^{r} l_i = m-d, \sum_{i=1}^{r} L_i = n-d\Big\}.$$

4.3.1.2 Structural properties of a LP ending with a last arrival event occurring during vacation period (SPLAEV)

Given non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(s \geq 1), l_1, l_2 \ldots, l_r$; $L_1, L_2, \ldots, L_{r-1}$; $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(s \geq 1), l_1, l_2 \ldots, l_r; L_1, L_2, \ldots,$ L_{r-1}, consider a lattice path under a $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches s times the barrier $Y = X$ never crosses it. Such a LP satisfies the following structural properties:

(a) $m - d$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq \text{Max}(k_0, L_1), l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - d$,

(b) $n - d$ vertical steps form $r - 1$ runs of lengths $L_1, L_2, \ldots, L_{r-1}$, respectively, satisfying $L_1, L_2, \ldots, L_{r-1} > 0$ and $\sum_{i=1}^{r-1} L_j = n - d$,

(c) $m - d$ horizontal steps and $n - d$ vertical steps form s closed cycles. Let $(1, u_1, u_2, \ldots, u_s)$ be the indices of the first AR in each cycle (with left ends on $Y = X$). Furthermore, $(l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s - 1}, l_r)$ are the lengths of the corresponding AR where ($l_i \geq k,$ for $i = 1, u_1, u_2, \ldots, u_{s-1}$ and $l_r < k$), satisfying $\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j$, $v = 0, 1, \ldots, s - 1$,

(d) The cycle at the end which is an open cycle made up of 1 AR of length l_{u_s}, satisfying $l_{u_s} < k, L_{u_s} = 0$,

(e) d_h diagonals are inserted each in any d_h out of $r - 1$ ARs, including the vertices at both ends of the runs except first k_0 vertices of first AR, first k vertices of u_i^{th} AR, $i = 1, 2, \ldots, s - 1$, and all vertices of r^{th} AR .

The conditions (a)-(d) above would be referred to as structural properties of a LP ending with a last arrival event under a vacation period (SPLAEV) of IBP of the system under $(0, k)$ policy.

Theorem 4.3.2. (The number of LPs satisfying to SPLAEV)

For non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$, $\underset{\sim}{L}^* = (l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_{r-1})$ let $LP_2^1\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}^*\big)$ denote the number of LP$_s$ satisfying the SPLAEV. Furthermore, the remaining $d - d_h$ diagonals are inserted each at any $n - d - r + 1$ vertices available along DR.

Then, for $k \geq 1,\ m \geq k_0 + k + 1,\ n \geq k_0, r \geq 2$ and $s \geq 1$,

$$LP_2^1\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}^*\big) = \sum_{(R_8^1, R_9^1)} \begin{pmatrix} n - d - r + 1 \\ d - d_h \end{pmatrix} \tag{4.3}$$

where

$$R_8^1 = \Big\{ \big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r - 1 \Big\},$$

$$R_9^1 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle_1 \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h \Big\},$$

$$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_{s-1}), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_{s-1}). \end{cases}$$

Proof. We follow generic steps used for proofing Theorem (4.3.1) to prove the above theorem. Let us consider a LP from $(0, 0)$ to $(m, n), m > n$. Delete all the d diagonal steps, and concatenate the steps to form a skeleton path consisting of AR and DR as described by vector $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$ and satisfying the structural conditions (a) to (e) SPLAEV.

Let d_h diagonals be inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, each of lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d - d_h$ diagonals can be inserted into any $d - d_h$ vertices out of $n - d - r + 1$ internal vertices available in the DR. This can be accomplished in $\begin{pmatrix} n - d - r + 1 \\ d - d_h \end{pmatrix}$ ways. Now summing $\begin{pmatrix} n - d - r + 1 \\ d - d_h \end{pmatrix}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ defined by R_8 and all insertion points $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ defined by R_9, we get (4.3). □

Lemma 4.3.2. Let $LP_2^1(k_0, k, m, n; d, s)$ be the number of LP$_s$ under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches s times the barrier $Y = X$, never crosses the line $Y = X$ and ends with a last arrival event occurring during vacation period, then summing (4.3) over r, d_h, and L^* we find

$$LP_2^1(k_0, k, m, n; d, s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_2^1\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big) \tag{4.4}$$

where

$$R_5^1 = \{r : 1 \le r \le \max(n - d - k_0 + 1, 2)\}$$
$$R_6^1 = \{d_h : \max(0, 2d - n + r - 1) \le d_h \le \min(r, d)\}$$
$$R_7^1 = \Big\{\underset{\sim}{L}^* : l_1 \ge \mathrm{Max}(k_0, L_1 + 1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \ge k, v = 0, 1, \ldots, s-1,\ l_{u_s} \ge \max\big(k, L_{u_s}\big), \sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i, \sum_{i=1}^{r} l_i = m - d, \sum_{i=1}^{r} L_i = n - d, \Big\}$$

4.3.1.3 Structural properties of a LP ending with a last arrival event occurring during busy period (SPLAEB)

Given non-negative integers $k_0, k(k \ge 1), m, n; d, d_h; r(r \ge 2), s(\ge 1), l_1, l_2 \ldots, l_r; L_1, L_2, \ldots, L_{r-1}$, consider a lattice path of a $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches the line $Y = X$ s times but does not cross it. Such a LP satisfies the following structural properties:

(a) $m - d$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \ge \mathrm{Max}(k_0, L_1), l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - d$,

(b) $n - d$ vertical steps form r runs of lengths $L_1, L_2, \ldots, L_{r-1}$, respectively, satisfying $L_1, L_2, \ldots, L_{r-1} > 0$ and $\sum_{i=1}^{r-1} L_j = n - d$,

(c) $m - d$ horizontal steps and $n - d$ vertical steps form s closed cycles. Let $(1, u_1, u_2, \ldots, u_s)$ be the indices of the first AR in each cycle (with left ends on $Y = X$). Furthermore, $(l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s})$ are the lengths of the corresponding AR where $l_i \geq k$, for $i = 1, u_1, u_2, \ldots, u_{v_s}$ satisfying $\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j$, $v = 0, 1, \ldots, s-1$,

(d) The cycle at the end which is an open cycle made up of $r - u_s + 1$ AR of lengths $l_{u_s}, l_{u_s+1}, \ldots, l_r$ respectively, satisfying $l_{u_s} \geq \text{Max}\big(k, L_{u_s}\big)$, $\sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i$,

(e) d_h diagonals are inserted each in any d_h out of r horizontal runs, including the vertices at both ends of the runs except first k_0 vertices of first AR and first k vertices of u_i^{th} AR, $i = 1, 2, \ldots, s$.

The conditions above would be referred to as structural properties of a LP ending with last arrival event occurring during busy period (SPLAEB) of IBP of the system under $(0, k)$ policy. In this situations, we have to consider the last arrival event which can happen in two cases i.e. under phase 1 or under phase 2 of service. The following two Theorems present the number of LPs corresponding to SPLAEB under phase 1 and phase 2 of service, respectively.

Theorem 4.3.3. (The number of LPs satisfying SPLAEB when a last arrival event occurring during phase 1 of service)
Given non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$, $\underset{\sim}{L}^* = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$ let $LP_2^2\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big)$, denote the number of LPs satisfying SPLAEB. Furthermore, the remaining $d - d_h$ diagonals are inserted each at any $n - d - r + 1$ vertices available along the DR.
Then, for $k \geq 1$, $m > k_0 + k + 1$, $n > k_0$, $r \geq 2$ and $s \geq 1$

$$LP_2^2\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}^*\big) = \sum_{(R_8, R_9)} \binom{n - d - r + 1}{d - d_h} \tag{4.5}$$

where

$$R_8 = \Big\{\big(i_1, i_2, \ldots, i_{d_h}\big)\colon 1 \leq i_1 < i_2 < \ldots < i_{d_h} < r\Big\},$$
$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big)\colon \triangle_1 \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$$
$$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s). \end{cases}$$

Proof. We follow generic steps used for proofing Theorem (4.3.1) to prove the above theorem. Let us consider a LP from $(0,0)$ to $(m,n), m > n$. Delete all the d diagonal steps, and concatenate the steps to form a skeleton path consisting of AR and DR as described by vector $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$ and satisfying the structural conditions (a) to (e) SPLAEB.

Under this condition i.e. the last arrival has to occur in phase 1 of service, the insertion of any diagonal should not take place in last AR. Hence, $i_{d_h} < r$. Let d_h diagonals be inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, each of lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d - d_h$ diagonals can be inserted into any $d - d_h$ vertices out of $n - d - r + 1$ internal vertices available in the DR. This can be accomplished in $\begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix}$ ways. Now summing $\begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ defined by R_8 and all insertion points $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ defined by R_9, we get (4.5). □

Lemma 4.3.3. Let $LP_2^2(k_0, k, m, n; d, s)$ be the number of LP$_s$ under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0,0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches the line Y=X s times, never crosses the line $Y = X$ and ends with a last arrival event occurring during phase 1 of service. Then, summing (4.5) over r, d_h, and L^* we find

$$LP_2^2(k_0, k, m, n; d, s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_2^2\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big) \tag{4.6}$$

where

$$R_5^1 = \{r : 1 \le r \le \max(n - d - k_0 + 1, 2)\}$$
$$R_6^1 = \{d_h : \max(0, 2d - n + r - 1) \le d_h \le \min(r, d)\}$$
$$R_7^1 = \Big\{\underset{\sim}{L}^* : l_1 \ge \text{Max}(k_0, L_1 + 1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \ge k, v = 0, 1, \ldots, s-1,\ l_{u_s} \ge \max\big(k, L_{u_s}\big), \sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i, \sum_{i=1}^{r} l_i = m - d, \sum_{i=1}^{r-1} L_i = n - d, \Big\}$$

Theorem 4.3.4. (The number of LPs satisfying SPLAEB when last arrival event occurring during phase 2 of service)
For non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s, s(s \geq 1)$, $\underset{\sim}{L^*} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$ let $LP_2^3\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L^*}\big)$, denote the number of LP$_s$ satisfying SPLAEB. Furthermore, the remaining $d - d_h$ diagonals are inserted each at any $n - d - r + 1$ vertices available along the DR.
Then, for $k \geq 1$, $m > k_0 + k + 1$, $n > k_0$, $r \geq 2$ and $s \geq 1$

$$LP_2^3\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L^*}\big) = \sum_{(R_8, R_9)} \begin{pmatrix} n - d - r + 1 \\ d - d_h \end{pmatrix} \tag{4.7}$$

where

$$R_8 = \Big\{ \big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{d_h}, i_{d_h} = r \Big\},$$

$$R_9 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle_1 \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h - 1, p_{i_v} \neq l_{i_v}, v = d_h \Big\},$$

$$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s). \end{cases}$$

Proof. We follow generic steps used for proofing Theorem (4.3.1) to prove the above theorem. Note that the insertion of a diagonal out of $d - d_h$ should take place in last AR but not at its right end vertex. The rest of the proof can be done by following to the proof of Theorem 4.3.3. □

Lemma 4.3.4. Let $LP_2^3(k_0, k, m, n; d, s)$ be the number of LP$_s$ under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches the line $Y = X$ s times, never crosses the line $Y = X$, and ends with a last arrival event occurring during phase 2 of service. Then, summing (4.7) over r, d_h, and L^* we find

$$LP_2^3(k_0, k, m, n; d, s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_2^3\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L^*}\big). \tag{4.8}$$

4.3.1.4 Structural properties of a LP ending with a last diagonal event (SPLDgE)

The structural properties of IBP of $M/C_2/1$ system under $(0, k)$ control policy when the system ends with a diagonal preceded by a departure (an arrival) is same with SPLDE (SPLAEB).

Theorem 4.3.5. (The number of LPs satisfying SPLDgE when the LP ends with a diagonal preceded by a departure)
For non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1), \underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$, let $LP_3^1\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}\big)$ denote the number of LP$_s$ satisfying the SPLDE. Furthermore, one diagonal is inserted at the end point of the last departure run and the remaining $d - d_h - 1$ diagonals are inserted each at any $n - d - r$ vertices available along the vertical runs.
Then for $r \geq 1$, $k \geq 1$, $m \geq k_0 + k + 1$, $n \geq k_0 + 1$, $r \geq 2$ and $s \geq 1$

$$LP_3^1\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}\big) = \sum_{(R_8, R_9)} \begin{pmatrix} n - d - r \\ d - d_h - 1 \end{pmatrix} \tag{4.9}$$

where

$$R_8 = \Big\{\big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r\Big\},$$

$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$$

$$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s). \end{cases}$$

Proof. To prove this theorem, we may follow to the proof of Theorem 4.1. Out of $d - d_h - 1$ diagonals need to be inserted, one diagonal should be inserted at the end of lattice path. Therefore, the remaining $d - d_h - 1$ diagonals have to be inserted in $n - d - r$. Hence the proof is complete and we get (4.9). □

Lemma 4.3.5. Let $LP_3^1(k_0, k, m, n; d; s)$ be the number of LP$_s$ under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches $Y = X$ s times but never crosses the line $Y = X$, and ends with a diagonal preceded by a departure, then summing (4.9) over r, d_h, and $\underset{\sim}{L}$

we find

$$LP_3^1(k_0, k, m, n; d; s) = \sum_{(R_5, R_6, R_7)} LP_3^1\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}\big) \tag{4.10}$$

where

$$R_5 = \{r : 2 \leq r \leq \max(n - d - k_0, 2)\},$$
$$R_6 = \{d_h : \max(0, 2d - n + r) \leq d_h \leq \min(r, d)\},$$
$$R_7 = \Big\{\underset{\sim}{L} : l_1 \geq \mathrm{Max}(k_0, L_1 + 1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k, v = 0, 1, \ldots, s-1, \sum_{i=u_s}^{r} l_i > \sum_{i=u_s}^{r} L_i, \sum_{i=1}^{r} l_i = m-d, \sum_{i=1}^{r} L_i = n-d\Big\}.$$

Theorem 4.3.6. (The number of LPs satisfying SPLDgE when the LP ends with a diagonal preceded by an arrival)

For non-negative integers $k_0, k(k \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$, $\underset{\sim}{L}^* = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$, let $LP_3^2\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big)$ denote the number of LP$_s$ satisfying the SPLAEB. Furthermore, one diagonal is inserted at the end point of the last arrival run and the remaining $d - d_h - 1$ diagonals are inserted each at any $n - d - r$ vertices available along the vertical runs.

Then for $r \geq 1$, $k \geq 1$, $m \geq k_0 + k + 1$, $n \geq k_0 + 1$, $r \geq 2$ and $s \geq 1$

$$LP_3^2\big(k_0, k, m, n; d, d_h, r, s, \underset{\sim}{L}^*\big) = \sum_{(R_8, R_9)} \begin{pmatrix} n - d - r + 1 \\ d - d_h \end{pmatrix} \tag{4.11}$$

where

$$R_8 = \Big\{\big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_r \leq r\Big\},$$
$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h, p_{i_{d_h}} = l_r, i_{d_h} = r\Big\},$$

$$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s). \end{cases}$$

Proof. We may follow the proof of Theorem 4.3.3 for proofing this theorem. □

Lemma 4.3.6. Let $LP_3^2(k_0, k, m, n; d; s)$ be the number of LP$_s$ under $(0, k)$ control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps (including those from $(0, 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches the line $Y = X$ s times, never crosses the line $Y = X$, and ends with a diagonal preceded by an arrival. Then, summing (4.11) over r, d_h, and L we find

$$LP_3^2(k_0, k, m, n; d; s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_3^2\big(k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big) \tag{4.12}$$

where

$R_5^1 = \{r : 2 \leq r \leq \max(n - d - k_0 + 1, 2)\}$,

$R_6^1 = \{d_h : \max(0, 2d - n + r) \leq d_h \leq \min(r, d)\}$

$R_7^1 = \{\underset{\sim}{L}^* : l_1 \geq \mathrm{Max}(k_0, L_1 + 1), \sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k, v = 0, 1, \ldots, s-1,\ l_{u_s} \geq \max\big(k, L_{u_s}\big), \sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i, \sum_{i=1}^{r} l_i = m - d, \sum_{i=1}^{r} L_i = n - d, \}$

For the case $r \geq 2$ and $d = d_h = 0$, we get

$$\begin{aligned} &LP_1(k_0, k, m, n; 0, s) = LP_2^1(k_0, k, m, n; 0, s) = LP_2^2(k_0, k, m, n; 0, s) \\ &= LP_2^3(k_0, k, m, n; 0, s) = LP_3^1(k_0, k, m, n; 0, s) = LP_3^2(k_0, k, m, n; 0, s) \\ &= \frac{m - n + k_0 + (s-1)k}{m + n - k_0 - (s-1)k} \binom{m + n - k_0 - (s-1)k}{m} \end{aligned} \tag{4.13}$$

4.3.2 Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ control policy

Let $f^i_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ system starts initially with k_0 customers at time t under $(0, k)$ control policy for the respective scenarios given by the index $i = 1, ..., 8$ in Table 4.3. Note that k_0 denotes the initial number of customers in the system when the system opens.

In the following theorems we repeatedly use the following notation. Given a IBP of length t we partition t into three portions τ, t_1 and t_2 denoting respectively the total periods of vacation, phase 1 service and phase 2 service such that $\tau = t_1 + t_2$ and $t_2 \geq 0$. Excluding the number of transitions from phase 1 service to phase 2 service, the number of transitions during IBP of length t is partitioned into N_0, N_1 and N_2 denoting the number of transitions during τ, t_1 and t_2 respectively.

Table 4.3: The scenarios of LP representation of IBP of $M/C_2/1$

Index i	**Scenario during period (0,t) and the relevant LP with the specified ending event**	**Theorem**
1	No phase 2 service and LP has end event during vacation period	-
2	No phase 2 service and the relevant LP has ending event in busy period	-
3	The corresponding LP satisfies SPLDE with last departure event	4.3.1
4	The corresponding LP satisfies SPLAEV with last arrival event under vacation	4.3.2
5	The corresponding LP satisfies SPLAEB with last arrival event under phase 1 of service	4.3.3
6	The corresponding LP satisfies SPLAEB with last arrival event under phase 2 of service	4.3.4
7	The corresponding LP satisfies SPLDE with last diagonal event preceded by a departure	4.3.5
8	The corresponding LP satisfies SPLAEB with last diagonal event preceded by an arrival	4.3.6

The structural sets and parameters used in defining pdf is presented in Table 4.4

Table 4.4: The structural sets and parameters used in defining pdf

Notation	Description
R_1	$= \{m : k_0 + k \leq m < \infty\}.$
R_1^1	$= \{m : k_0 + k + 1 \leq m < \infty\}.$
R_1^*	$= \{k^* : 0 \leq k^* \leq k - 1\}.$
R_1^2	$= \{m : k_0 + k + 2 \leq m < \infty\}.$
R_1^3	$= \{m : k_0 + k + 3 \leq m < \infty\}.$
R_2	$= \{n : k_0 \leq n < m - k^*\}.$
R_2^1	$= \{n : k_0 + 1 \leq n \leq m - 1\}.$
R_2^2	$= \{n : k_0 + 2 \leq n \leq m - 1\}.$
R_2^3	$= \{n : k_0 + 3 \leq n \leq m - 1\}.$
R_3	$= \left\{s : 1 \leq s < \left\lfloor \frac{n-k_0}{k} + 1 \right\rfloor \right\}.$
R_4	$= \left\{d : 1 \leq d \leq \min\left(\lfloor \frac{n}{2} \rfloor, m - k_0\right)\right\}$ where $[x]$ denotes the largest integer contained in x.
R_5	$= \{r : 2 \leq r \leq \max(n - d - k_0, 2))\}.$
R_5^1	$= \{r : 2 \leq r \leq \max(n - d - k_0 + 1, 2))\}.$
R_6	$= \{d_h : \max(0, 2d - n + r) \leq d_h \leq \min(r, d)\}.$
R_6^1	$= \{d_h : \max(0, 2d - n + r - 1) \leq d_h \leq \min(r, d)\}.$
R_7	$= \Big\{ \underset{\sim}{L}^* : l_1 \geq \text{Max}(k_0, L_1 + 1), \sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k$ $v = 0, 1, \ldots, s-1, \sum_{i=u_s}^{r} l_i > \sum_{i=u_s}^{r} L_i,$ $\sum_{i=1}^{r} l_i = m - d, \sum_{i=1}^{r} L_i = n - d \Big\}.$
R_7^1	$= \Big\{ \underset{\sim}{L} : l_1 \geq \text{Max}(k_0, L_1 + 1), \sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k$ $v = 0, 1, \ldots, s-1,\ l_{u_s} \geq \max\big(k, L_{u_s}\big), \sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i,$ $\sum_{i=1}^{r} l_i = m - d, \sum_{i=1}^{r} L_i = n - d \Big\}.$
R_8	$= \Big\{ \big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r \Big\}.$
R_8^1	$= \Big\{ \big(i_1, i_2, \ldots, i_{d_h}\big) \colon 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r - 1 \Big\}.$
R_9	$= \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h \Big\},$ $\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_{s-1}), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_{s-1}). \end{cases}$

Notation	Description
R_9^1	$= \left\{ p_i = (p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}) \colon \triangle_1 \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h \right\}$, $\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s - 1), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s - 1). \end{cases}$
α_1	: the probability that a customer will move from phase 1 to the phase 2.
β	: the probability that a customer will leave the system completely after completing phase 1 of service.
λ	: the average rate of customers entering the queueing system.
μ_1	: the average rate of serving customers in phase 1.
μ_2	: the average rate of serving customers in phase 2.

Theorem 4.3.7. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ control policy such that no customer receives phase 2 service and system ends in vacation period)
Let $f^1_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ system starting initially with k_0 customers at time t under $(0, k)$ policy without encountering phase 2 service and ending in vacation period. Then we have

$$f^1_{k_0;k}(t) = e^{-(\lambda+\mu_1)t} \sum_{(R_1, R_1^*, R_2, R_3)} \frac{k_0 + (s-1)k}{2n - k_0 - (s-1)k} \binom{2n - k_0 - (s-1)k}{n} \times \lambda^{m-k_0} \frac{\beta^n}{\Gamma(k^* + (s-1)k + 1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma(k^* + (s-1)k + i + 1)}{\Gamma(m + n - k_0 + i + 1)} \tag{4.14}$$

where $R_1, R_1^*, R_2, R_3, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. The above equation corresponds to the case when all customers depart after having phase 1 service and system ends in vacation period, i.e. after the s^{th} (last) touch occurs, there are $k^* < k$ horizontal steps up to (m, n) where $m > n$ and $m = n + k^*$, $k^*(< k)$, $k^* = 0, 1, 2, \ldots, k - 1$.

To prove the identity (B.3), first we need to count the number of lattice paths under scenario 1 of Table 4.3. For counting the lattice paths, let $AB_1 \ldots B_s B$ be a lattice path as shown in Figure 4.3 stipulated by structural properties for last arrival event (SPLAEV) such that after s^{th} (last) touch occurs, there are $k^* < k$ horizontal steps up to (m, n) where $m > n$ and $m = n + k^*, k^* < k$, in which the path from $(k_0, 0)$ to (m, n) touches s times the line $Y = X$ at the points $B_1, \ldots, B_s$.

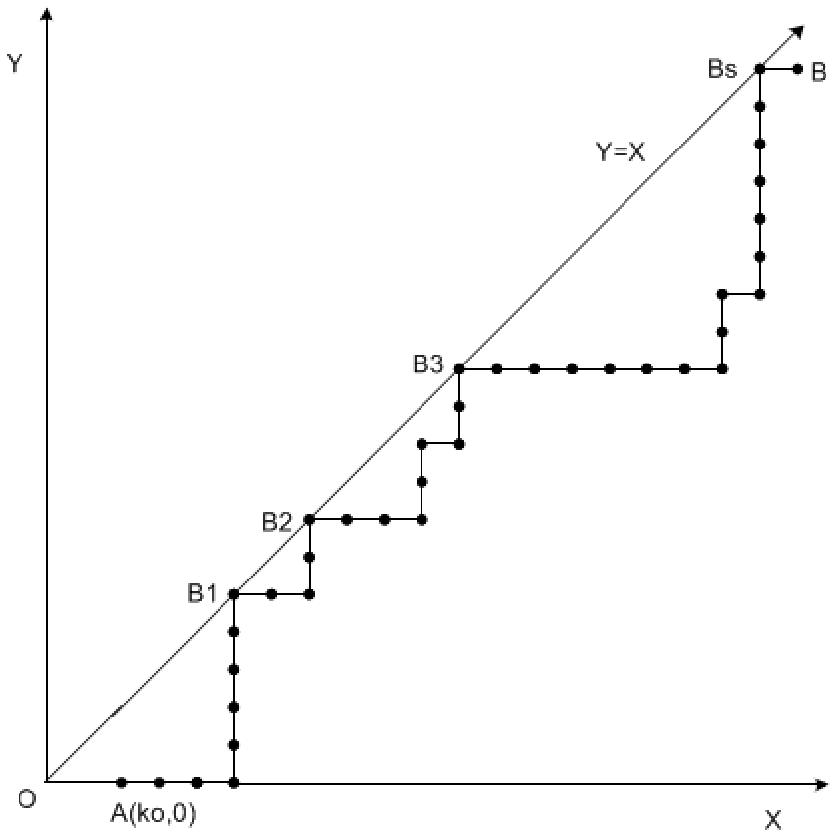


Figure 4.3: A lattice path stipulated by SPLAEV

First, ignore the last segment B_sB. Then, we have the set of paths from $(k_0, 0)$ to (n, n) having s touches the line $Y = X$ at the points $B_1, \ldots, B_s$. A 1:1 mapping is constructed by removing the first k horizontal steps from $B_1B_2, \ldots, B_{s-1}B_s$. We obtain a 1:1 set of paths from $(k_0, 0)$ to $(n - (s - 1)k, n)$ which do not touch the line $Y = X + (s - 1)k$. The number of paths is given by

$$\frac{k_0 + (s-1)k}{2n - k_0 - (s-1)k} \binom{2n - k_0 - (s-1)k}{n}. \tag{4.15}$$

Next, let $t = \tau + t_1$ be the total time spent in the system, where τ is the total time spent in vacation periods and t_1 is the total time spent in phase 1 of service.

The number of arrivals is $m - k_0$. This consists of $(s-1)k + k^*$ arrivals during the vacation periods and $m - (s-1)k - k^* - k_0$ arrivals during the busy periods. The number of departures is n. Therefore, the total number of transitions is $m + n - k_0$. Call this as N_0.

Let $\tau = \tau_1 + \tau_2 + \ldots + \tau_s$ be the the total vacation period in the system where τ_i is the length of the i^{th} vacation period in the i^{th} cycle, $(i = 1, 2, \ldots, s)$ of the corresponding LP. The transitions in each τ_i for $i = 1, 2, \ldots, s-1$ consist of k arrivals while the transition in τ_s consist of k^* arrivals. The number of arrivals is $k^* + (s-1)k$. The probability density function of τ is $s(k^*k) - Erlang$ with parameter λ given by

$$f_{s(k^*k)}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{(s-1)k+k^*}}{\Gamma((s-1)k + k^* + 1)}$$

as the result of

$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}\lambda^k\tau_y^{k-1}}{\Gamma(k)} \frac{e^{-\lambda\tau_y}\lambda^{k^*}\tau_s^{k^*-1}}{\Gamma(k^*)} d\tau_s d\tau_y =$$

$$\frac{e^{-\lambda\tau}\lambda^{(s-1)k}\lambda^{k^*}}{(\Gamma(k))^{s-1}\Gamma(k^*)} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \tau_y^{k-1}\tau_s^{k^*-1} d\tau_s d\tau_y$$

The derivation of the probability density function above is provided in Appendix B.1.

The total number of transitions in $t_1 = t - \tau$, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as explained below:

Number of arrivals: $m - (s-1)k - k^* - k_0$ since $(s-1)k + k^*$ is the number of arrivals while system is in idle period.

Number of departures after phase 1: n.

Note that the total number of transitions during the period t_1 is

$$N_1 = m + n - (s-1)k - k^* - k_0.$$

The probability density function of $t_1 = t - \tau$ is $N_1 - Erlang$ with parameter $(\lambda + \mu_1)$ given by

$$f_{N_1}(t-\tau) = \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda + \mu_1)^{N_1}(t-\tau)^{N_1-1}}{\Gamma(N_1)}.$$

The density function of this IBP is given by

$$
\begin{aligned}
f^1_{k_0;k}(t) = & \sum_{(R_1,R^*,R_2,R_3)} \frac{k_0+(s-1)k}{2n-k_0-(s-1)k} \begin{pmatrix} 2n-k_0-(s-1)k \\ n \end{pmatrix} \\
& \times \int_{\tau=0}^{t} \frac{e^{-\lambda\tau}(\lambda\tau)^{(s-1)k+k^*}}{\Gamma((s-1)k+k^*+1)} \\
& \times \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{m+n-(s-1)k-k^*-k_0}(t-\tau)^{m+n-(s-1)k-k^*-k_0-1}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
& \times \left(\frac{\lambda}{\lambda+\mu_1}\right)^{m-(s-1)k-k^*-k_0} \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n} d\tau.
\end{aligned}
$$

The integral part can be simplified into

$$
\lambda^{m-k_0} \frac{\beta^n e^{-(\lambda+\mu_1)t}}{\Gamma((s-1)k+k^*+1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma((s-1)k+k^*+i+1)}{\Gamma(m+n-k_0+i+1)}. \quad (4.16)
$$

The simplification of the integral part is given in (B.1) in Appendix B.2.

Substituting (B.1) in Appendix B.2 into the above function we obtain the density function for this incomplete busy period at time t, i.e.

$$
\begin{aligned}
f^1_{k_0;k}(t) = & e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_1^1,R_2,R_3)} \frac{k_0+(s-1)k}{2n-k_0-(s-1)k} \begin{pmatrix} 2n-k_0-(s-1)k \\ n \end{pmatrix} \\
& \times \lambda^{m-k_0} \frac{\beta^n}{\Gamma(k^*+(s-1)k+1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma((s-1)k+k^*+i+1)}{\Gamma(m+n-k_0+i+1)}.
\end{aligned}
$$

□

Theorem 4.3.8. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy with no customer receiving phase 2 service and the system ends in busy period)

Let $f^2_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under $(0, k)$ policy where system starts initially with k_0 customers, has no phase 2 service and ends in busy period. Then we have

$$f_{k_0}^2(t) = e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_2,R_3)} \left\{ \begin{pmatrix} m+n-k_0-sk \\ n \end{pmatrix} - \begin{pmatrix} m+n-k_0-sk \\ m \end{pmatrix} \right\} \times \lambda^{m-k_0} \frac{\beta^n}{\Gamma(sk+1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma(sk+i+1)}{\Gamma(m+n-k_0+i+1)} \tag{4.17}$$

where where $R_1, R_2, R_3, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. The above equation corresponds to the case when all customers depart after having phase 1 service, and so in the corresponding LP we have $d = 0$. To prove the identity (4.17), first we need to count the number of lattice paths under the scenario 2 of Table 4.4.

For counting the lattice paths, let $OAB_1 \ldots B_sB$ be a lattice path (Figure 4.4) envisaged in structural properties for last arrival event (SPLAEB) such that after s^{th}(last) touch occurs, there are $k^*(< k)$ horizontal steps up to (m, n) where $m > n$ and $m \geq n + k^*$, note that in the last open cycle which is in busy period, at every step number of horizontal steps is $>$ number of vertical steps and/or stipulated in structural properties for last departure event (SPLDE) in which the path from $(k_0, 0)$ to (m, n) touches s times the line $Y = X$ at the points $B_1, \ldots, B_s$.

A 1:1 mapping is constructed by removing the first k horizontal steps from $B_1B_2, \ldots, B_{s-1}B_s, B_sB$ and getting the set of paths from $(k_0, 0)$ to $(m - sk, n)$ which always remain below the line $Y = X + sk$ and do not even touch it in-between. Then, for $k_0 > sk \geq 0, m > n \geq 0, m \geq k_0$, the number of paths is given by

$$\begin{pmatrix} m+n-k_0-sk \\ n \end{pmatrix} - \begin{pmatrix} m+n-k_0-sk \\ m \end{pmatrix}. \tag{4.18}$$

Next, let $t = \tau + t_1$ be the total time spent in the system, where τ is the total time in vacation state and t_1 the total time spent in phase 1 of service.

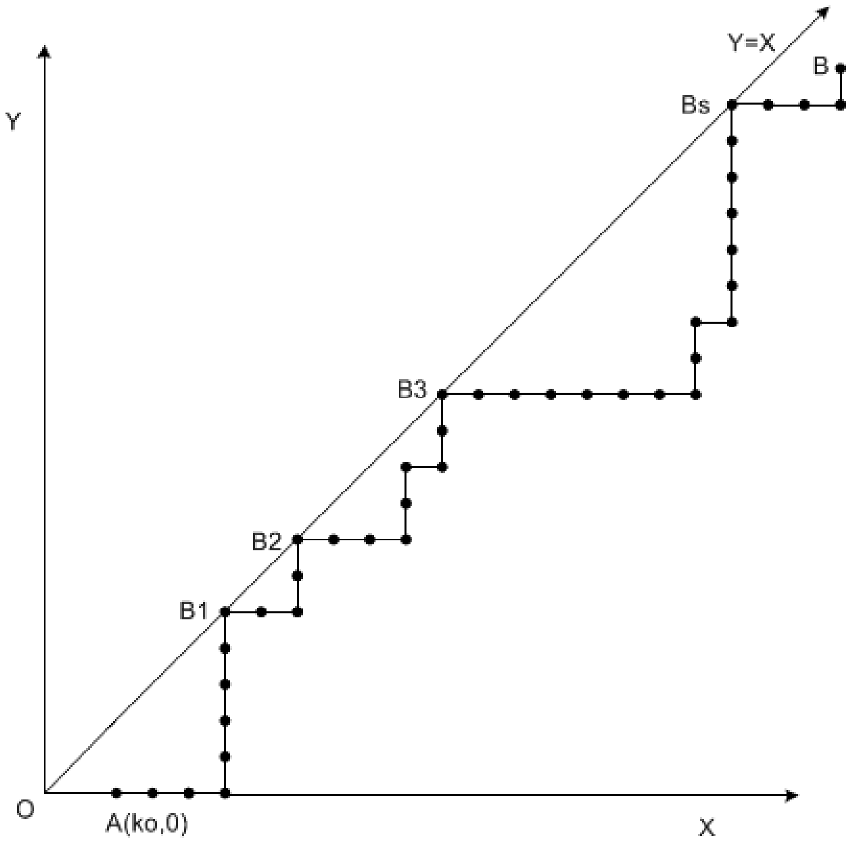


Figure 4.4: A lattice path stipulated by SPLAEB

The number of arrivals is $m - k_0$. This consists of sk arrivals during the vacation periods and $m - sk - k_0$ arrivals during the busy periods. The number of departures is n. Therefore total number of transitions is $m + n - k_0$. Call this as N_0.

Let $\tau = \tau_1 + \tau_2 + \ldots + \tau_s$, be the total time system is in vacation period where τ_i is the time system length of the i^{th} vacation period in the i^{th} cycle $(i = 1, 2, \ldots, s)$ of the corresponding LP. The transitions in each τ_i consist of k arrivals. The number of arrivals is sk. The probability density function of τ is $s(k^*k) - Erlang$ with parameter λ given by

$$f_{sk}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}$$

as the result of

$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda^k \tau_y} \lambda^k \tau_y^{k-1}}{\Gamma(k)} d\tau_y = \frac{e^{-\lambda\tau_y} \lambda^{sk}}{(\Gamma(k))^s} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s} \tau_y^{k-1} d\tau_y.$$

The derivation of the probability density function above is provided in Appendix B.3.

The total number of transitions in $t_1 = t - \tau$, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as explained below:

Number of arrivals: $m - k_0 - sk$ since sk is the number of arrivals while system is in vacation period.

Number of departures after phase 1: n.

The total number of transitions during the phase 1 service time t_1 is

$$N_1 = m + n - k_0 - sk.$$

The probability density function of $t_1 = t - \tau$ is $N_1 - Erlang$ with parameter $(\lambda + \mu_1)$ given by

$$f_{N_1}(t-\tau) = \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{N_1}(t-\tau)^{N_1-1}}{\Gamma(N_1).}$$

The probability density function for this incomplete busy period is

$$\begin{aligned} f^2_{k_0;k}(t) = & \sum_{(R_1,R_2,R_3)} \left\{ \binom{m+n-k_0-sk}{n} - \binom{m+n-k_0-sk}{m} \right\} \\ & \times \int_{\tau=0}^{t} \frac{e^{-\lambda\tau}\lambda^{s(k-1)}\tau^{sk}}{\Gamma(sk+1)} \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{m+n-k_0-sk}(t-\tau)^{m+n-k_0-sk-1}}{\Gamma(m+n-k_0-sk)} \\ & \times \left(\frac{\lambda}{\lambda+\mu_1}\right)^{m-k_0-sk} \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n} d\tau \end{aligned}$$

Substituting (B.2) in Appendix B.4 into the above function, hence, we obtain the probability density function $f^2_{k_0;k}(t)$ as below.

$$\begin{aligned} f^2_{k_0;k}(t) = & e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_2,R_3)} \left\{ \binom{m+n-k_0-sk}{n} - \binom{m+n-k_0-sk}{m} \right\} \\ & \times \lambda^{m-k_0} \frac{\beta^n}{\Gamma(sk+1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma(sk+i+1)}{\Gamma(m+n-k_0+i+1)}. \end{aligned}$$

□

Theorem 4.3.9. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy ending with last departure event)

Let $f^3_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under $(0, k)$ policy where system starts initially with k_0 customers and ends with last departure event. Then we have

$$\begin{aligned} f^3_{k_0;k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^1,,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \alpha_1^d \beta^{n-2d} \\ & \times \lambda^{m-d-k_0} \mu_1^{n-d} \sum_{i=0}^{\infty} \frac{\mu_2^{d+i}}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\ & \times \frac{\Gamma(sk+i+1)}{\Gamma(sk+1)} \frac{\Gamma\big(m+n-2d-k_0-sk+j-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-sk-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}, \end{aligned} \tag{4.19}$$

where $R_1^1, R_2^1, R_3, R_4, R_5, R_6, R_7, R_8, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. The RHS of (4.19) corresponds to the scenario 3 of Table 4.3, that is the density of the IBP such that the system still busy while ending with a departure either after phase 1 or phase 2 of service.

Let $t = \tau + t_1 + t_2$ be the total time spent in the system, where τ is the total vacation time, t_1 is the total time spent in phase 1 of service and $t_2 = t - \tau - t_1$ is the total time spent in phase 2 of service.

For fixed $\underset{\sim}{L}$ of a LP of a IBP of length t, total number of transitions when the system is still active is given by $m + n - d - k_0$ as described below:

Number of arrivals during the vacation period: sk.

Number of arrivals during the busy period: $m - sk - d - k_0$.

Number of departures after phase 1: $n - 2d$.

Number entries into phase 2: d.

Number of departures after phase 2: d.

The number of transitions in τ, the total time system is in vacation, consists of only arrivals. The number of arrivals is sk. The probability density function of

τ with parameter λ is given by (as in Theorem 4.3.6)

$$f_{sk}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}.$$

The total number of transitions in t_1, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as well as entries into phase 2 of service as explained below:

Number of arrivals: $m - d - k_0 - sk - \sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$ since sk is the number of arrivals while system is in vacation period and $\sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$ is the number of arrivals while customers are in phase 2 of service.

Number of entries into phase 2: d.

Number of departures after phase 1: $n - 2d$.

The total number of transitions during phase 1 service time t_1 is

$$N_1 = m + n - 2d - k_0 - sk - \sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big).$$

The probability density function of phase 1 service time t_1 is $N_1 - Erlang$ with parameter $(\lambda + \mu_1)$ given by

$$f_{N_1}(t_1) = \frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1} t_1^{N_1-1}}{\Gamma(N_1)}.$$

The total number of transitions in $t_2 = t - \tau - t_1$, the time spent in phase 2 of service, consists of arrivals and departures while customers are in phase 2 of service as explained below:

Number of arrivals: $\sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$.

Number of departures: d.

Therefore N_2, total number of transitions during $t_2 = t - \tau - t_1$ will be

$$N_2 = \sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big) + d.$$

The probability density function of phase 2 service time $t_2 (= t - \tau - t_1)$ is $N_2 - Erlang$ with parameter $(\lambda + \mu_2)$ given by

$$f_{N_2}(t - \tau - t_1) = \frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}(t-\tau-t_1)^{N_2-1}}{\Gamma(N_2)}$$

The derivation of (4.19) is provided in Appendix B.5.

Substituting (B.6) (in Appendix B.5) in (B.3) (in Appendix B.5) and for values of N_1, N_2, we get

$$\begin{aligned} f^3_{k_0;k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^1,,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \alpha_1^d \beta^{n-2d} \\ & \times \lambda^{m-d-k_0} \mu_1^{n-d} \sum_{i=0}^{\infty} \frac{\mu_2^{d+i}}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\ & \times \frac{\Gamma(sk+i+1)}{\Gamma(sk+1)} \frac{\Gamma\big(m+n-2d-k_0-sk+j-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-sk-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}. \end{aligned}$$

□

Theorem 4.3.10. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy such that last arrival event happened in vacation period)
Let $f^4_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (0,k) policy given that system starts initially with k_0 customers when the last arrival event is in phase 1 of service ending in vacation (corresponding to a case under the scenario 4 of Table 4.3). Then we have

$$\begin{aligned} f^4_{k_0}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R^2_1, R^*_1, R^1_2, R_3, R_4, R^1_5, R^1_6, R^1_7, R^1_8, R_{9^1})} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d} \\ & \times \lambda^{m-d-k_0} \mu_1^{n-d} \sum_{i=1}^{\infty} \frac{\mu_2^{d+i}}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\ & \times \frac{\Gamma((s-1)k+k^*+i+1)}{\Gamma((s-1)k+k^*)} \\ & \times \frac{\Gamma\big(m+n-2d-k_0-(s-1)k-k^*+j-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}{\Gamma\big(m+n-2d-k_0-(s-1)k-k^*-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}, \end{aligned} \tag{4.20}$$

where $R^2_1, R^*_1, R^1_2, R_3, R_4, R^1_5, R^1_6, R^1_7, R^1_8, R_{9^1}, \alpha, \beta, \lambda, \mu_1,$ and μ_2 are given in Table 4.4.

Proof. The identity in (4.20) corresponds to a case of scenario 4 of Table 4.3 when last event is an arrival which happened in phase 1 of service ending in vacation period. Note that an arrival event during phase 2 service will initiate the phase 1 service postponing the vacation period. Here $N_1 = m+n-2d-k_0-(s-1)k-k^*-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$ and $N_2 = \sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)-d$. In this case, the number of entries into phase 2 is d the number of departures after phase 2. The rest of the proof is similar to that of Theorem 4.3.9. Table 4.5 presents the summary of the transitions.

□

Table 4.5: Transitions for $M/C_2/1$ system under $(0, k)$ policy for vacation period, t_1 and t_2: the system ends with an arrival in vacation period

Event	**Vacation period**	t_1	t_2
Number of arrivals	$(s-1)k + k^*$	$m - d - k_0 - (s-1)k - k^*$ $- \sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$	$\sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$
Number of departure after phase 1	-	$n - 2d$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	d
Total number of transition		N_1	N_2
Erlang **distribution**	$\frac{e^{-\lambda\tau}\lambda^{s(k-1)}\tau^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}}{\Gamma(N_1)}$ $\times t_1^{N_1-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}}{\Gamma(N_2)}$ $\times (t-\tau-t_1)^{N_2-1}$

Theorem 4.3.11. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy when the last event is an arrival in phase 1 of service)

Let $f^5_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (0,k) policy such that system starts initially with k_0 customers and the last arrival event is in phase 1 of service (corresponds to scenario 5 of Table 4.3). Then we have

$$
\begin{aligned}
f^5_{k_0;k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^2, R_2^1, R_3, R_4, R_5^1, R_6^1, R_7^1, R_8, R_9)} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d} \\
& \times \lambda^{m-d-k_0} \mu_1^{n-d} \times \sum_{i=1}^{\infty} \frac{\mu_2^{d+i}}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\
& \times \frac{\Gamma(sk+i+1)}{\Gamma(sk)} \frac{\Gamma\big(m+n-2d-k_0-sk+j-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}{\Gamma\big(m+n-2d-k_0-sk-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)},
\end{aligned}
\tag{4.21}
$$

where $R_1^2, R_2^1, R_3, R_4, R_5^1, R_6^1, R_7^1, R_8, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. The identity (4.21) corresponds to the case when the last event is an arrival which happened in phase 1 of service (scenario 5 of Table 4.4). Here $N_1 = m + n - 2d - k_0 - sk - \sum_{v=1}^{d_h}$ and $N_2 = \sum_{v=1}^{d_h}$. Table 4.6 presents the summary of the transitions.

Table 4.6: Transitions for $M/C_2/1$ system under $(0, k)$ policy for vacation period, t_1 and t_2: the system ends with an arrival in phase 1

Event	**Vacation period**	t_1	t_2
Number of arrivals	sk	$m - d - k_0 - sk$ $- \sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$	$\sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$
Number of departure after phase 1	-	$n - 2d$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	d
Total number of transition	sk	N_1	N_2
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}}{\Gamma(N_1)}$ $\times t_1^{N_1-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}}{\Gamma(N_2)}$ $\times(t-\tau-t_1)^{N_2-1}$

In this case, the number of entries into phase 2 is d the number of departures after phase 2. The further proof is similar to the proof of Theorem 4.3.9. □

Theorem 4.3.12. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy when the last event is an arrival in phase 2 of service)

Let $f^6_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ system starts initially with k_0 customers and the last arrival event is in phase 2 of service corresponding to scenario 6 of Table 4.3. Then we have

$$
\begin{aligned}
f^{6}_{k_0;k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^3,R_2^2,R_3,R_4,R_5^1,R_6^1,R_7^1,R_8,R_9)} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d+1} \\
& \times \lambda^{m-d-k_0} \mu_1^{n-d+1} \sum_{i=1}^{\infty} \frac{\mu_2^{d+i+1}}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\
& \times \frac{\Gamma(sk+i+1)}{\Gamma(sk)} \frac{\Gamma\big(m+n-2d-k_0-sk+j+1-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-sk+1-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)},
\end{aligned}
\tag{4.22}
$$

where $R_1^3, R_2^2, R_3, R_4, R_5^1, R_6^1, R_7^1, R_8, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. The identity (4.22) corresponds to the case when last event is arrival which happened in phase 2 of service which implies the number of entries into phase 2 is d while the number of departures after phase 2 is $d-1$. Here $N_1 = m+n-2d-k_0-sk-\sum_{v=1}^{d_h}$ and $N_2 = \sum_{v=1}^{d_h} + d - 1$. Table 4.7 presents the summary of the transitions. The further proof is similar to the proof of Theorem 4.3.11. □

Table 4.7: Transitions for $M/C_2/1$ system under $(0, k)$ policy for vacation period, t_1 and t_2: the system ends with an arrival in phase 2

Event	**Vacation period**	t_1	t_2
Number of arrivals	sk	$m - d - k_0 - sk$ $- \sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$	$\sum_{v=1}^{d_h}\big(l_{i_v} - p_{i_v}\big)$
Number of departure after phase 1	-	$n - 2d$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	$d - 1$
Total number of transition	sk	N_1	N_2
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}}{\Gamma(N_1)}$ $\times t_1^{N_1-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}}{\Gamma(N_2)}$ $\times(t-\tau-t_1)^{N_2-1}$

Theorem 4.3.13. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy ending with a diagonal preceded by a departure)

Let $f^7_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (0,k) policy such that system starts initially with k_0 customers and ends with a diagonal preceded by a departure corresponding to scenario 7 of Table 4.3. Then we have

$$\begin{aligned} f^7_{k_0;k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h-1 \end{pmatrix} \alpha^d \beta^{n-2d+1} \\ & \lambda^{m-d-k_0}\mu_1^{n-d+1} \times \sum_{i=1}^{\infty}\frac{\mu_2^{d+i+1}}{i!}\sum_{j=1}^{\infty}\frac{(\mu_2-\mu_1)^j}{j!}\frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\ & \times \frac{\Gamma(sk+i+1)}{\Gamma(sk)} \frac{\Gamma\big(m+n-2d-k_0-sk+j+1-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}{\Gamma\big(m+n-2d-k_0-sk+1-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}, \end{aligned} \tag{4.23}$$

where $R_1^2, R_2^3, R_3, R_4, R_5, R_6, R_7, R_8, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. Equation (4.23) corresponds to the case when the density of the system is still active when the system ends with a diagonal preceded by a departure, which can occur after phase 1 as well as after phase 2 of service.

To prove this term, let $t = \tau + t_1 + t_2$ be the total time spent in the system, where τ is the total time in idle, t_1 the total time spent in phase 1 of service and t_2 the total time spent in phase 2 of service. In this case the number of entries into phase 2 is d as the number of departures after phase 2. The further proof is similar to the proof of Theorem 4.3.9. Table 4.8 presents the summary of the transitions. □

Table 4.8: Transitions for $M/C_2/1$ system under $(0, k)$ policy for vacation period, t_1 and t_2: the system ends with a diagonal preceded by a departure

Event	**Vacation period**	t_1	t_2
Number of arrivals	sk	$m - d - k_0 - sk$ $- \sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$	$\sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$
Number of departure after phase 1	-	$n - 2d + 1$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	$d - 1$
Total number of transition	sk	N_3	N_4
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_3}}{\Gamma(N_3)}$ $\times t_1^{N_3-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_4}}{\Gamma(N_4)}$ $\times(t-\tau-t_1)^{N_4-1}$

Theorem 4.3.14. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ policy ending with a diagonal preceded by an arrival)

Let $f^8_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under $(0, k)$ policy such that system starts initially with k_0 customers and ends with a diagonal preceded by a departure corresponding to scenario 8 of Table 4.3. Then we have

$$
\begin{aligned}
f^8_{k_0;k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d+1} \\
& \lambda^{m-d-k_0} \mu_1^{n-d+1} \times \sum_{i=1}^{\infty} \frac{\mu_2^{d+i+1}}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\
& \times \frac{\Gamma(sk+i+1)}{\Gamma(sk)} \frac{\Gamma\big(m+n-2d-k_0-sk+j+1-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-sk+1-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)},
\end{aligned}
\tag{4.24}
$$

where $R_1^2, R_2^3, R_3, R_4, R_5, R_6, R_7, R_8, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 4.4.

Proof. The identity (4.24) corresponds to the density of the system is still active when the system ends with diagonal preceded by an arrival occurred in phase 1 of service only.

To prove the above equation, let $t = \tau + t_1 + t_2$ be the total time spent in the system, where τ is the total time in idle, t_1 the total time spent in phase 1 of service and t_2 the total time spent in phase 2 of service. Table 4.9 presents the summary of the transitions.

In this case the number of shifting into phase 2 of service is d and the the number of departures after phase 2 is $d - 1$. The further proof is similar to the proof of Theorem 4.3.11. □

Table 4.9: Transitions for $M/C_2/1$ system under $(0, k)$ policy for vacation period, t_1 and t_2: the system ends with a diagonal preceded by an arrival

Event	**Vacation period**	t_1	t_2
Number of arrivals	sk	$m - d - k_0 - sk$ $- \sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$	$\sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$
Number of departure after phase 1	-	$n - 2d + 1$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	$d - 1$
Total number of transition	sk	N_3	N_4
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_3}}{\Gamma(N_3)}$ $\times t_1^{N_3-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_4}}{\Gamma(N_4)}$ $\times(t-\tau-t_1)^{N_4-1}$

Combining theorems 4.3.7., 4.3.8., 4.3.9., 4.3.10., 4.3.11., 4.3.12., 4.3.13, 4.3.14 we have the following main theorem.

Theorem 4.3.15. (Probability density function of IBP of $M/C_2/1$ system under $(0, k)$ control policy)

Let $f_{k_0;k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under $(0, k)$ policy given that system starts initially with k_0 customers. Then we have

$$f_{k_0;k}(t) = \sum_{i=1}^{8} f^i_{k_0;k}(t). \tag{4.25}$$

4.4 Numerical Computations and Comments

In this section, we present the numerical computation of equation (4.25). For the purpose of the computation, an R programme has been developed. The corresponding code can be found in Appendix A2. The program starts with generating all possible lattice paths using the library AlgDesign. Next, only the paths satisfying the SPLDE, SPLAEV, SPLAEB and SPLDgE are filtered. These paths form the set L and finally equation (4.25) is computed for the selected paths.

The code for computing the density is presented in Appendix A2. On a computation process for $t = 0.03(0.01)4$, the computation takes < 6 CPU minutes.

Numerical computations have been performed for different parameters involved. The outputs are presented in Figure 4.5 - Figure 4.8. From these graphs, for $0 < t \leq 4$ we notice that as t increases the density function of incomplete busy period $f_{k_0}(t)$ computed for $t = 0.03(0.01)4$ increases, then decreases after attaining a maximum value, i.e. satisfying the expected pattern at a larger unit of time we expect the density function to cease, leading to zero probability at such time i.e. $t = 4$, thus justifying our results.

From Figure 4.5, for different values of λ we notice that as t increases, the density function of IBP system increases. Then after attaining a maximum value decreases. The rate of increase in the value of density for smaller values of λ is less than that for the larger values of λ.

It can be seen from Figure 4.6 that for different values of μ_1 we notice that as t increases, the density function of IBP system increases. Then, after attaining a maximum decreases. The rate of increase for smaller values of μ_1 is less than for larger values of μ_1 and the rate of decrease for smaller values of μ_1 is less than for larger values of μ_1.

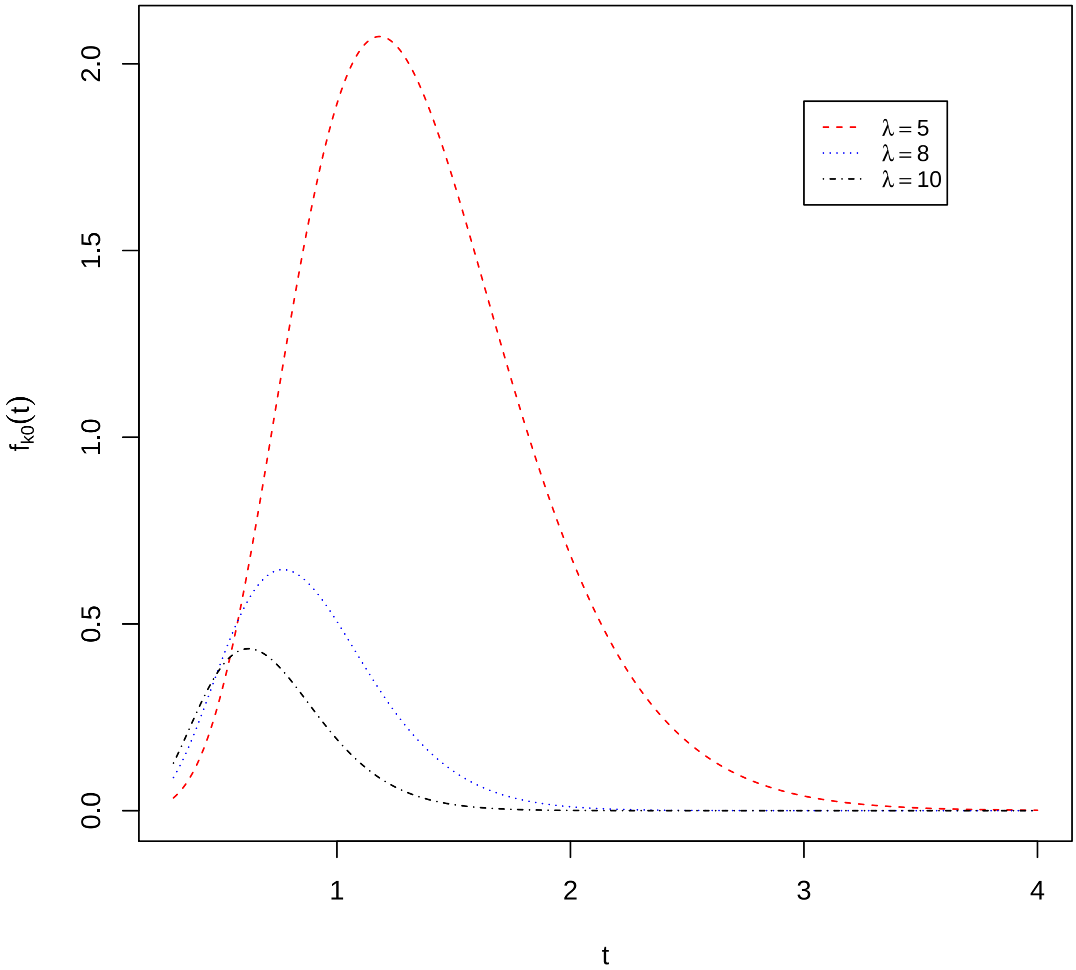


Figure 4.5: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k}(t)$, for different λ taking $k_0 = 1,\ k = 2,\ \mu_1 = 8,\ \mu_2 = 4,\ \alpha = 0.4,\ \beta_1 = 0.6$

Figure 4.7 shows that for different values of μ_2, as t increases, the density function of IBP system increases and after attaining a maximum, it decreases. As μ_2 increases, the density also increases. But for a particular value of t, we notice that IBP system increases with a decrease in the value of μ_2.

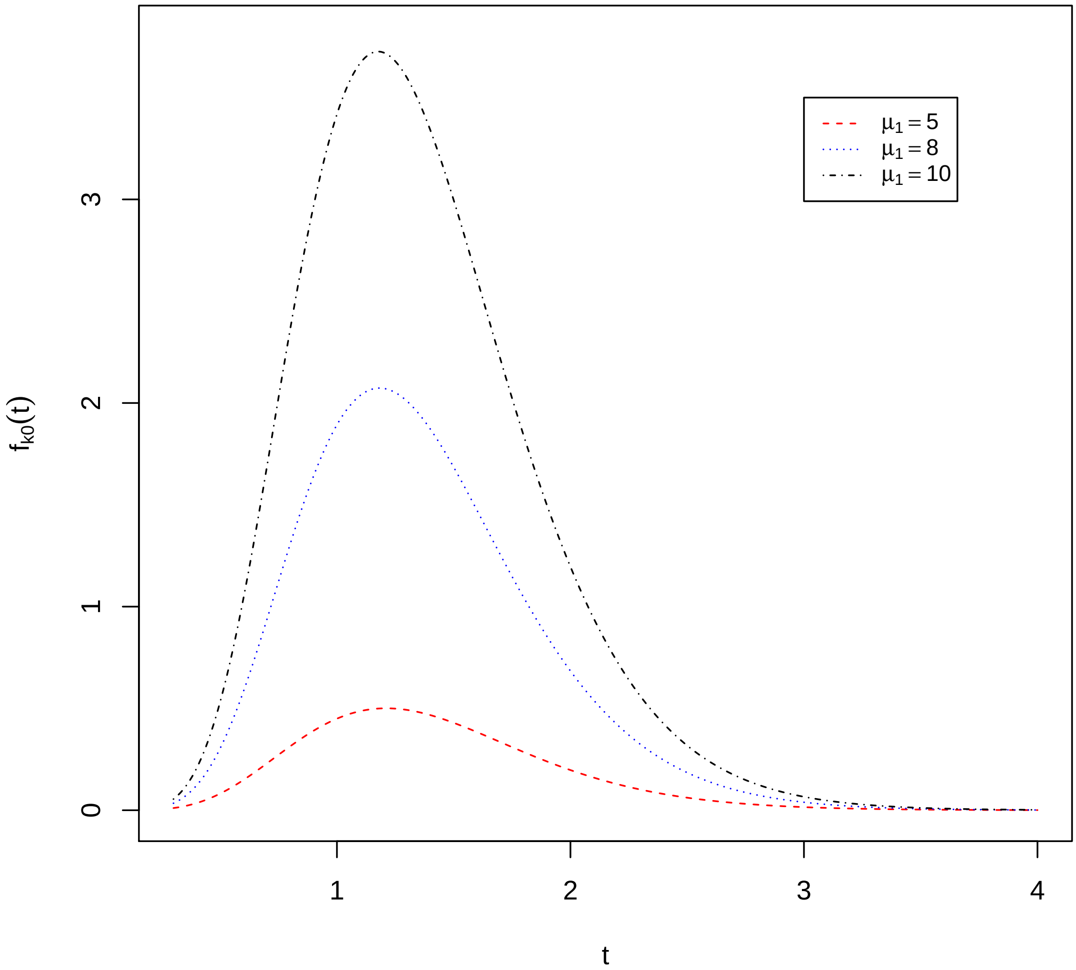


Figure 4.6: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k}(t)$, for different μ_1 taking $k_0 = 1,\ k = 2,\ \lambda = 5,\ \mu_2 = 4,\ \alpha = 0.4,\ \beta_1 = 0.6$

Figure 4.8 shows that for different values of α and β, as t increases, the density function of IBP system initially increases, then decreases. An increase in α means more and more customers enter the next phase of service. The rate of decrease in the value of density for smaller values of α is greater than that for the larger value of α.

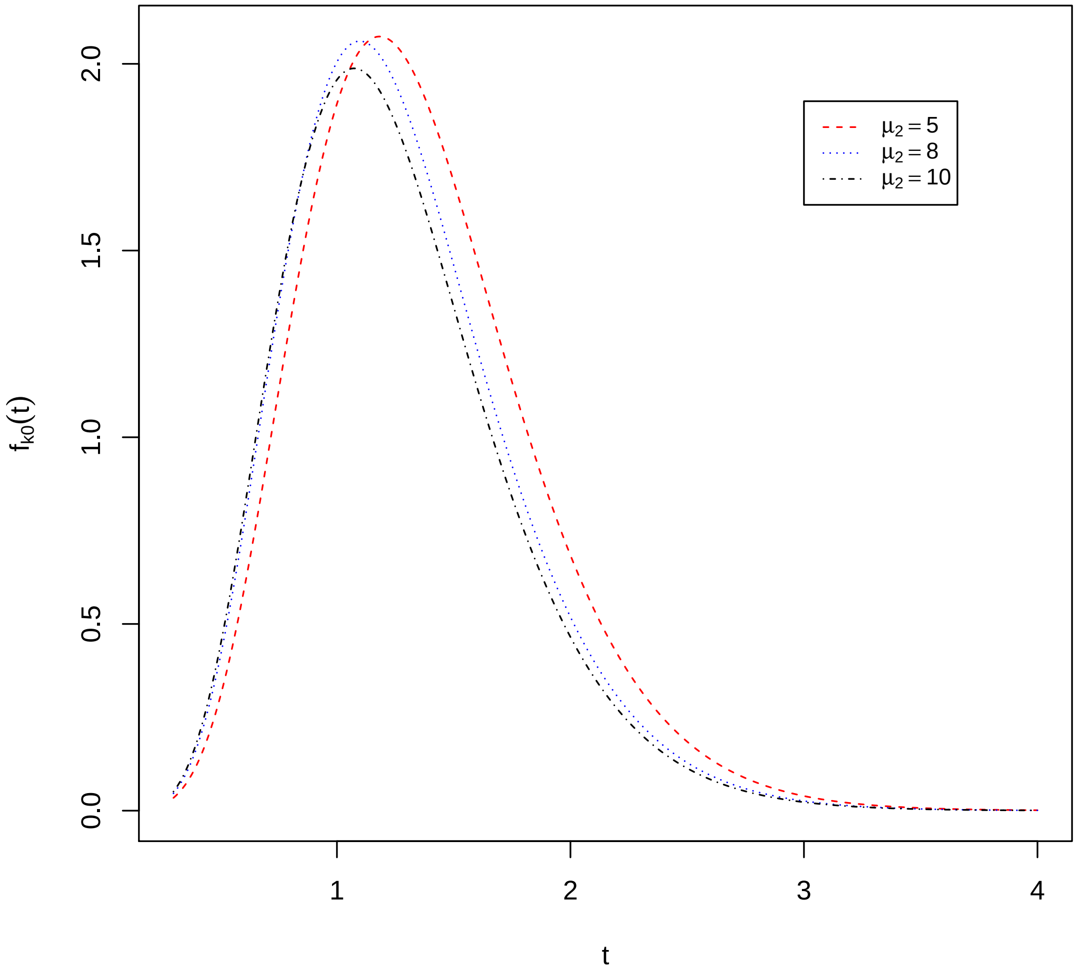


Figure 4.7: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k}(t)$, for different μ_2 taking $k_0 = 1,\ k = 2,\ \lambda = 5,\ \mu_1 = 8,\ \alpha = 0.4,\ \beta_1 = 0.6$

4.5 Summary

In this chapter, we constructed and computed the incomplete busy period density function of a $M/G/1$ queues under $(0, k)$ control policy using the LPC approach, which entails approximating general service distribution by Coxian distribution.

The number of lattice paths given that system ends with a last departure event, a last arrival event occurring during vacation period, a last arrival event occurring during phase 1 or phase 2 of service,a last diagonal event preceded by a departure and a last diagonal event preceded by a departure are presented in equations

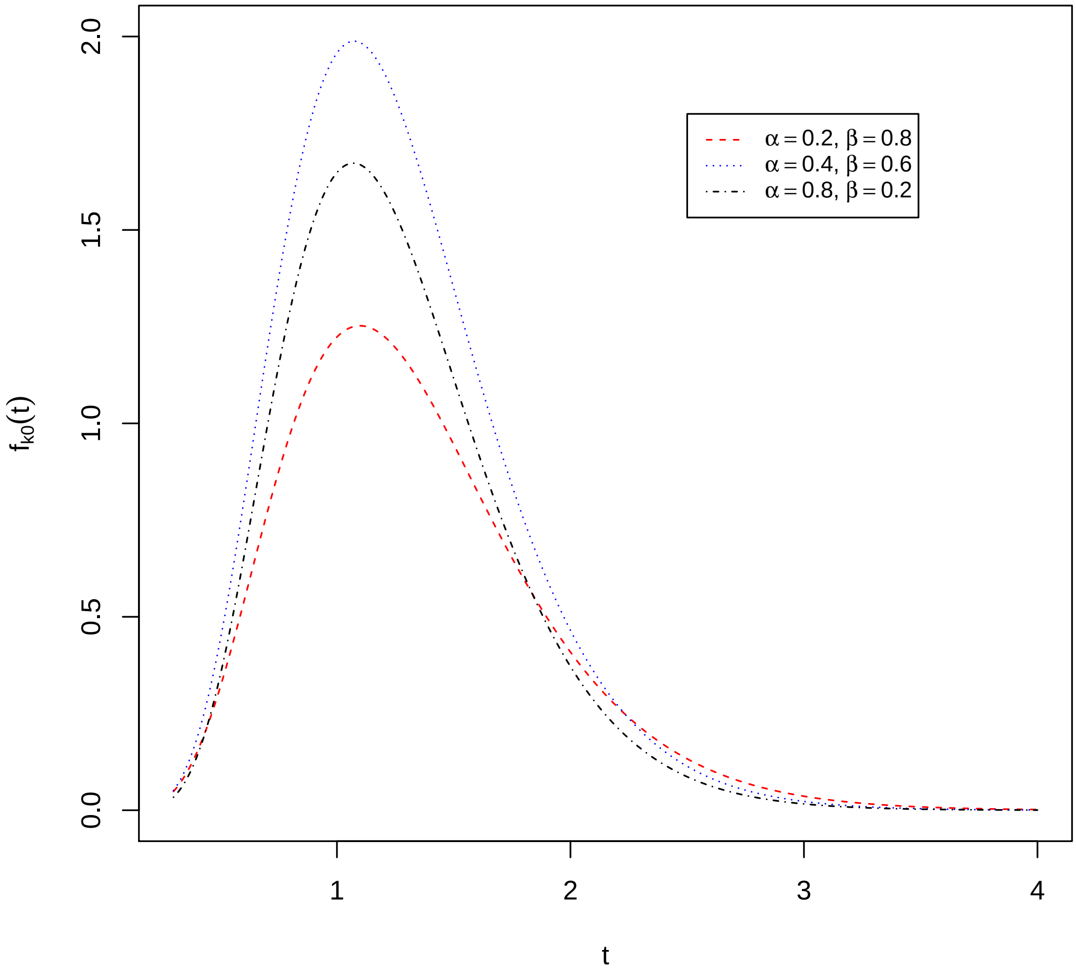


Figure 4.8: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k}(t)$, for different α and β taking $k_0 = 1,\ k = 2,\ \lambda = 5,\ \mu_1 = 8,\ \mu_2 = 4$

(4.1), (4.3), (4.5), (4.7), (4.9) and (4.11), respectively. Using LPC results and keeping track of number of transition in each phase, we develop the explicit form of transient analysis of the incomplete busy period is given in (4.25).

Finally, we develop the computer codes to estimate the busy period densities for the above models. This includes generating all possible lattice paths using the library AlgDesign. Next, only the paths satisfying conditions of the structural properties of are filtered. These paths form the set L and finally the density function is derived for the selected paths. The required algorithms and the computational procedures are developed as R program (Team [97]).

We study the sensitivity of the analysis of IBP for various parameters setting in Figure 4.5 - Figure 4.8. We observe that for given parameter setting, as t increases, the density function of IBP system $f_{k_0}(t)$ computed for $t = 0.03(0.01)4$ decreases.

Chapter 5

M/G/1 QUEUE UNDER $(k^{'}, k)$ CONTROL POLICY

5.1 General

Many real life queueing systems are operated under control policies determined heuristically with the objective of optimal management of the system. In this chapter, we focus on the (k', k) control policy. Under this policy the server starts serving only when the number of customers in the queue becomes k and remains busy as long as there are at least k' customers waiting for service. Several authors have investigated (k', k) control policy for Markovian queues. For example Sen et al. [89], Böhm and Mohanty [7] have derived transient solution $M/M/1$ queueing system under (k', k) control policy using lattice path approach. Moreover, Ke et al. [49] considered (k', k) policy for $M/M/c$ queue. However, results are not available on the probability density function of IBP of queuing systems with general service under (k', k) control policy.

In this chapter we focus on studying the time dependent behavior queuing systems of type $M/G/1$ operating under (k', k) policy. Note that the number of customers in the queue during a vacation period is always greater than or equal to k' and less than k. During a busy period the number of customers in the queue is always greater than k'. We allow an arbitrary number of, say, k_0 customers initially waiting. Here in, we compute the closed form expression for incomplete busy period of such queueing process through LPC.

Motivated by the previous work, we approximate general service by Coxian 2-phase distribution. Under this approximation, $M/G/1$ system is equivalent to $M/C_2/1$ system, that retains Markovian structure.

The rest of the chapter is organized as follows. In the following section, we describe the problem and mathematical formulation to determine the density function of IBP of $M/G/1$ queueing system under (k', k) policy. Then in section 5.3, we present the results on counting the number of lattice paths and subsequent computation of transient probabilities of IBP. In section 5.4, we carry out some numerical computations to investigate the performance of the density for different parameters involved. Finally, summary is given in section 5.5.

5.2 The $M/C_2/1$ Model under $(k^{'}, k)$ Control Policy

We recapitulate the methodology described in Section (3.2) to derive the pdf of the IBP of $M/G/1$ queueing system under $(k^{'}, k)$ policy. As in the previous chapter, see Section (4.2), $M/G/1$ model will be approximated as $M/C_2/1$. That is general service is approximated as Coxian 2-phase model. For the purpose of obtaining the expression for the transient probabilities of the IBP, we have to consider:

(i) The set of lattice paths in which after s^{th} (last) touch of the barrier $Y = X - k^{'}$, the lattice path ends in vacation period with $k^*(< k - k^{'})$ horizontal steps up to (m, n) where $m > n$ and $m = n + k^*$, $k^* < k - k'$, $k^* = 0, 1, 2, \ldots, (k - k' - 1)$.

(ii) The set of lattice paths in which after each of the s touches with the barrier $Y = X - k^{'}$ the path has at least $k - k^{'}$ horizontal steps preceding the first vertical step (or no vertical step for the last touch). In this case, we find the system ends in busy period.

The queueing process is observed at the points of transitions over of time interval $(0, t)$. At this stage, an arrival, a departure or shifting of a customer to phase 2 on the completion of phase 1 service is represented by a unit of horizontal, unit of vertical or $\sqrt{2}$ unit of diagonal steps, respectively.

In a lattice path representation of a $M/G/1$ queueing system, a vertex (x, y) has the following interpretation:

$x =$ the sum of number of arrivals, number of initial customers and number of customers shifting to phase 2 service,

$y =$ the sum of number of departures and number of customers shifting to phase 2 service.

Thus, in the lattice path we have $x \geq y$, in other words lattice path will never cross the barrier $Y = X - k^{'}$ in the plane, but may touch it.

Let $f_{k_0;k',k}(t)$ denote the pdf of IBP of $M/C_2/1$ under $(k^{'}, k)$ control policy, for $t \geq 0$, where k_0 is the initial number of customers in the system, k' is the number of customers in the system when server stops the service i.e starts vacation and k is initial number of customers at the start of busy period and/or the minimum number of customers in the queue required for the system to start the busy period. The queueing process of $M/C_2/1$ during a IBP may be represented by a lattice path in a two dimensional plane starting at $(k_0, 0)$ and ending at (m, n). Such a lattice path may touch the barrier $Y = X - k^{'}$, say s times. After every touch there must be at least k customers before service can start. The IBP of a queueing system with $k^{'} = 3$, $k_0 = 5$ and $k = 7$ is represented by lattice path, as illustrated in Figure 5.1. In the graph of the lattice path we represent an arrival or departure occurring during the progress of phase 1 (phase 2) service by solid (dotted) horizontal or solid (dotted) vertical line. Furthermore, a shift to phase 2 service from phase 1 is denoted by solid diagonal line in the graph of LP. Note that in Figure 5.1 server is active (in serving mode) at the end point of the path.

Such a lattice path (corresponding to a IBP during $(0, t)$ with k_0 initial customers in the system) will either end with a vertical step (departure), a horizontal step (arrival) or a diagonal step (shift to phase 2 service).

The set of all lattice paths satisfying the properties of IBP period over time $(0, t)$ can be partitioned into three cases as follows:

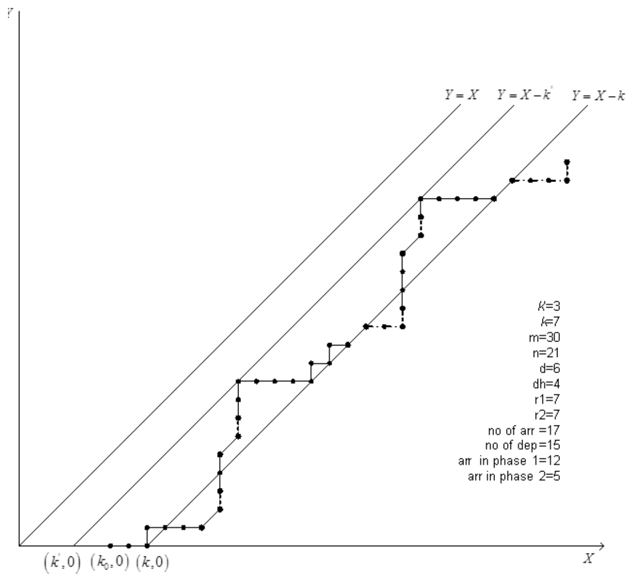


Figure 5.1: $M/C_2/1$ model. An example of lattice path representation under $(k^{'}, k)$ control policy

(i) The set of lattice paths ending with solid (or dashed) vertical line corresponding to a departure after phase 1 (or phase 2) service.

(ii) The set of lattice paths ending with solid (or dashed) horizontal line corresponding to an arrival during phase 1 (or phase 2) service.

(iii) The set of lattice paths ending with a solid diagonal line corresponding to a customer shifting to phase 2 of service. Note that such a solid diagonal line can occur immediately after a solid vertical line (a departure after phase 1 service), dotted vertical line (a departure after phase 2 service) or solid horizontal line (an arrival during the operation of phase 1 service).

Table 5.1 illustrates the decomposition of lattice paths into disjoint groups as explained above in (i)-(iii). The last column provides the reference to the theorem where the counting results on the corresponding lattice paths are established. Each of these steps is illustrated in details over the remaining sections.

Next, the pdf $f_{k0;k',k}(t)$ is estimated as probability corresponding to the number of lattice paths satisfying the properties of IBP over time $(0, t)$. This is accomplished by counting the number of lattice paths satisfying the properties of IBP in Theorems 5.4.1, 5.4.2, 5.4.3, 5.4.4, 5.4.5 and 5.4.6 corresponding to the ending structure of the lattice paths. Finally, the probabilities corresponding to such paths are computed using lattice paths corresponding to $M/C_2/1$ model to arrive at $f_{k_0;k',k}(t)$.

5.2.1 Transitions

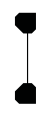
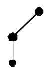
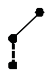
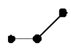
The staying time in each state is an exponential random variable with a parameter depending on the state given below:

$$P\{T_{n+1} - T_n > t | X_n = i\} = \begin{cases} e^{-(\lambda+\mu_u)t}, & \text{if a costumer is undergoing phase } u \\ & \text{of service, } u = 1, 2 \\ e^{-\lambda t}, & \text{if the system is under vacation.} \end{cases}$$

5.2.2 Counting of lattice paths

In this section, we present the process of counting lattice paths. We, first, construct a skeleton by removing all the diagonals of a LP in Figure 5.1 (See Figure 5.2). For the purpose of counting the definition AR and DR as in chapter 3 will be followed. Now, it can be seen from Figure 5.2 that this skeleton LP only consists of horizontal and vertical runs which represent arrivals in phase 1 and departures after phase 1, respectively. As we have already removed all diagonals from Figure 5.1, when we count the possible number of LPs, we need to consider all possibilities of inserting the diagonals into horizontal and/or vertical runs following the restrictions as presented on page (43-44). Furthermore, no diagonal can be inserted in the first k horizontal steps after a touch with the barrier.

Table 5.1: Structural properties of LPs ending event and its related theorems

System vacation/busy	**Description of ending event of LP**	**Representation**	**Theorem Reference**
Vacation period	- LP with last arrival event during vacation period		5.3.2
	- LP with last departure event after phase 1 of service		5.3.1
	- LP with last diagonal event preceded by departure after phase 1 of service		5.3.5
	- LP with last diagonal event preceded by departure after phase 2 of service		5.3.5
Busy period	- LP with last arrival event during phase 1 of service		5.3.3
	- LP with last diagonal event preceded by an arrival in phase 1		5.3.6
	- LP with last departure event during phase 2 of service		5.3.1
	- LP with last arrival event during phase 2 of service		5.3.4

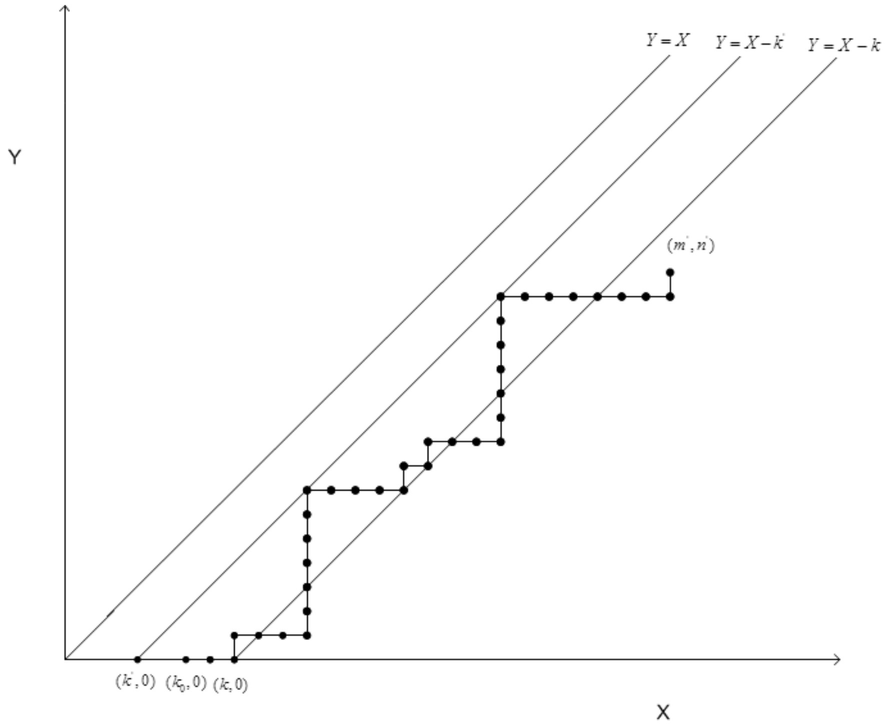


Figure 5.2: Skeleton path of an example of lattice path representation under $(k^{'}, k)$ control policy

5.2.3 Notation and terminology

In a lattice path, for $k'(k' \geq 1), k(k - k' \geq 1), n < m$, Table 5.2 presents the notation and terminology that we use in this chapter. Where possible, we have retained notations from previous chapters.

Table 5.2: The notation and terminology

Notation	Description
k_0	Initial number of customers at the start of busy period.
k	Initial number of customers at the start of busy period and/or the minimum number of customers in the queue required for the system to start the busy period.
k'	The number of customers in the system when server stops the service i.e starts vacation.
m	Total number of arrivals and customers shifting to phase 2 service.
n	Total number of departures and customers shifting to phase 2 service.
s	The number of times a LP from $(k_0, 0)$ to (m, n), $m > n$ touches the line $Y = X - k'$ giving rise to s AR with left ends on the line $Y = X - k'$ $(s \geq 1)$. A LP will never cross the line $Y = X - k'$.
r	Number of AR $(r \geq 2)$.
d	Total number of diagonals inserted in AR and/or DR $(d \geq 0)$ from $(k, 0)$ to (m, n).
d_k	Total number of diagonals inserted in AR $(d_k \geq 0)$ from $(k_0, 0)$ to (m, n).
$d_{k'}$	Total number of diagonals inserted in AR which start from level k' $(d_{k'} \geq 0)$ from $(k, 0)$ to (m, n).
$d - d_k$	Number of diagonals inserted in DR from $(k_0, 0)$ to (m, n).
$d - d_{k'}$	Total number of diagonals inserted in AR from $(k, 0)$ to (m, n) other than diagonals which start from level k'.
$l_{\underset{\sim}{i}}$	Lengths of horizontal runs $\underset{\sim}{i}$.
l_i	Length of the i^{th} AR $(i = 1, 2, \ldots, r)$.
L_i	Length of the i^{th} DR $(i = 1, 2, \ldots, r)$.
$\underset{\sim}{L}$	$(l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_r)$.
$\underset{\sim}{L}^*$	$(l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$.
$\underset{\sim}{i}$	$(i_1, i_2, \ldots, i_{d_h})$, d_h indices of the AR in each of which a diagonal is inserted.

Notation	**Description**
$\underset{\sim}{u}$	$(u_1, u_2, \ldots, u_s)$ the number of s runs horizontal starting from a contact point on the line $Y = X - k'$ or start from level k'.
$\underset{\sim}{u}^*$	$(u_s, u_s + 1, \ldots, r)$ the index number of AR after s^{th} touch.
$l_{\underset{\sim}{i}}$	$(l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}})$, lengths of AR $\underset{\sim}{i}$. $(u_1, \ldots, u_s) \subset (i_1, i_2, \ldots, i_{d_h})$.
$l_{\underset{\sim}{u}}$	$(l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s})$, lengths of AR $\underset{\sim}{u}$.
$l_{\underset{\sim}{u}^*}$	$(l_{u_s}, l_{u_s+1}, \ldots, l_r)$, lengths of AR $\underset{\sim}{u}^*$.
$p_{\underset{\sim}{i}}$	$(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}})$, distances from extreme left end points where diagonals are inserted in horizontal runs $\underset{\sim}{i}$ including vertices at both ends of the runs).
$CC(\underset{\sim}{u})$	s closed cycles: complete busy period of AR $\underset{\sim}{u}$.
$OC(\underset{\sim}{u}^*)$	1 opened cycles: pure incomplete busy period of AR $\underset{\sim}{u}^*$.

To illustrate these notations, we refer to Figure 5.1. Here $k' = 3,\ k_0 = 5,\ k = 7,\ s = 2,\ u_0 = 1,\ \underset{\sim}{u} : (3,6),\ \underset{\sim}{u}^* : (6),\ l_i : (7,3,4,1,3,7),\ L_i : (1,6,1,1,6,1),\ \underset{\sim}{i} : (2,5,6),\ l_{\underset{\sim}{u}} : (4,7), l_{\underset{\sim}{u}^*} : (6), p_{\underset{\sim}{i}} : (3,1)$. Hence, $CC(\underset{\sim}{u})$ is 2 closed cycles of AR $\underset{\sim}{u}$. These are

(1). $\sum_{j=1}^{2} l_j = \sum_{j=1}^{2} L_1$ where $\sum_{j=1}^{2} l_j = l_1 + l_2 = 7,\ \sum_{j=1}^{2} L_1 + L_2 = 7.$

(2). $\sum_{j=3}^{4} l_j = \sum_{j=3}^{4} L_1$ where $\sum_{j=3}^{5} l_j = l_3 + l_4 + l_5 = 8,\ \sum_{j=3}^{5} L_3 + L_4 + L_5 = 8.$

Here, $\underset{\sim}{u}^* : 5$, therefore, $OC(\underset{\sim}{u}^*)$ is 1 opened cycle of AR $\underset{\sim}{u}^*$ is $l_6 = 7 > L_6 = 1$.

5.3 Results on Lattice Path Counting

The following results of lattice paths have been established which will be used in the next section to obtain the expression for the probability density function of the $M/G/1$ system active under (k, k') policy.

5.3.1 Structural properties of lattice paths

In this section, we discuss the special features of lattice paths corresponding to the scenario described in Table 5.1 and use them to count the number of lattice paths.

5.3.1.1 Structural properties of a LP ending with a last departure event (SPLDE)

Given non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k'+1), k(k-k' \geq 1), m, n; d, d_h;$ $r(r \geq 2), s(\geq 1),\ l_1, l_2 \ldots, l_r;\ L_1, L_2, \ldots,\ L_r$, consider a lattice path under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m-d-k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n-d$ vertical steps and d diagonals steps such that the LP touches $Y = X - k'$ exactly s times never crosses the line $Y = X - k'$. Such a LP satisfies the following structural properties:

(a) $m-k'-d$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq \text{Max}(k_0 - k', L_1), l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - k' - d$,

(b) $n-d$ horizontal steps form r runs of lengths $L_1, L_2, \ldots, L_r$, respectively, satisfying $L_1, L_2, \ldots, L_r > 0$ and $\sum_{i=1}^{r} L_j = n - d$,

(c) $m-k'-d$ horizontal steps and $n-d$ vertical steps form s complete busy period cycles and an open cycle ending at (m, n). Let $(1, u_1, u_2, \ldots, u_s)$ be the indices of the first AR in each cycle (with left ends on $Y = X - k'$). Furthermore, $l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s}$ are the lengths of the corresponding AR where $l_1 \geq \text{Max}(k_0 - k', L_1)$, $l_i \geq k - k'$ for $i = u_1, u_2, \ldots, u_{u_s}$ satisfying $\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j$, $v = 0, 1, \ldots, s-1$,

(d) The last cycle at the end is an open cycle comprising of $r - u_s + 1$ AR of lengths $l_{u_s}, l_{u_s+1}, \ldots, l_r$, respectively, satisfying $l_{u_s} \geq \text{Max}(k - k', L_{u_s})$, $\sum_{i=u_s}^{r} l_i > \sum_{i=u_s}^{r} L_i$,

(e) d_h diagonals are inserted each in any d_h out of r ARs, including the vertices at both ends of the runs except for first $k - k'$ vertices of first AR and first $k_0 - k'$ vertices of u_i^{th} AR for $i = 1, 2, \ldots, s$.

The conditions (a)-(e) above would be referred to as structural properties of a LP ending with a last departure event (SPLDE) of IBP of the system under (k', k) policy.

Theorem 5.3.1. (The number of LPs satisfying SPLDE)
For non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k'+1), k(k-k' \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$, $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_r)$, let $LP_1\big(k', k_0, k; m, n; d, d_h; r, s, \underset{\sim}{L}\big)$ denote the number of LP$_s$ satisfying the SPLDE. Furthermore, the remaining $d - d_h$ diagonals are inserted each at any $n - d - r$ vertices available along the vertical runs (excuding the last vertex i.e. the end of the lat DR).
Then, for $k' \geq 1,\ k_0 \geq k'+1,\ k \geq k'+1,\ m \geq 2k - k',\ n \geq k - k' + 1,\ r \geq 2$ and $s \geq 1$

$$LP_1\big(k', k_0, k; m, n; d, d_h, r, s, \underset{\sim}{L}\big) = \sum_{(R_8, R_9)} \begin{pmatrix} n - d - r \\ d - d_h \end{pmatrix} \tag{5.1}$$

where

$$R_8 = \Big\{(i_1, i_2, \ldots, i_{d_h}) : 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r\Big\},$$

$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin \big(1, u_1, u_2, \ldots, u_s\big), \\ k_0 - k', & \text{if } i_v = 1, \\ k - k', & \text{if } i_v \in \big(u_1, u_2, \ldots, u_s\big). \end{cases}$$

Proof. Before proofing the above theorem, let us consider generic steps taken to prove the theorem. These are

(i) to consider a LP from $(0, 0)$ to (m, n),

(ii) to delete all the diagonal steps,

(iii) to concatenate the steps to form a skeleton path consisting of AR and DR satisfying the properties of respective structural properties,

(iv) to enumerate the number of LP by inserting all possible diagonals.

If we delete all the diagonal steps, and compress to form a skeleton path, then there must be $m - k' - d$ horizontal steps, and $(n - d)$ vertical steps. To get skeleton from $(k', 0)$ to $(m - k' - d, n - d)$, we suppose this skeleton consists of r ARs runs and r DRs of lengths $l_i (i = 1, 2, \ldots, r)$, and $L_j (j = 1, 2, \ldots, r)$ respectively. One unique path will be produced by this scenario. For the purpose of insertion, suppose d_h diagonals are inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, respectively with lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d - d_h$ diagonals will be inserted into any $d - d_h$ vertices out of $n - d - r$. The number to do this is $\begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix}$.

Now, summing $\begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ and $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$, we get (5.1). □

Lemma 5.3.1. Let $LP_1(k', k_0, k, m, n; d, s)$ be the number of lattice paths under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n - d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times, never crosses the line $Y = X - k'$ and ends with a last departure event. Then, summing (5.1) over r, d_h, and $\underset{\sim}{L}$ we find

$$LP_1(k', k_0, k, m, n; d, s) = \sum_{(R_5, R_6, R_7)} LP_1(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}) \qquad (5.2)$$

where

$$R_5 = \{r : 2 \leq r \leq \max(\min(m - k' - d, n - d), 2)\},$$
$$R_6 = \{d_h : \max(0, 2d - n + r) \leq d_h \leq \min(r, d)\},$$
$$R_7 = \Big\{\underset{\sim}{L} : l_1 \geq \mathrm{Max}(k - k', L_1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k - k', v = 0, 1, \ldots, s-1,\ 1 \leq s \leq r-1, \sum_{i=1}^{r} l_i = m - k' - d, \sum_{i=1}^{r} L_i = n - d\Big\}.$$

5.3.1.2 Structural properties of a LP ending with a last arrival event occurring during vacation period (SPLAEV)

Given non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k' + 1), k(k - k' \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1),\ l_1, l_2 \ldots, l_r; L_1, L_2, \ldots, L_{r-1}$, consider a lattice path under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d - k'$ horizontal steps (including those from $(k', 0)$ to $(k_0, 0)$), $n - d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times never crosses the line $Y = X - k'$. Such a LP satisfies the following structural properties:

(a) $m-k'-d$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq \text{Max}(k_0 - k', L_1), l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m - k' - d$,

(b) $n-d$ vertikal steps form r runs of lengths $L_1, L_2, \ldots, L_{r-1}$, respectively, satisfying $L_1, L_2, \ldots, L_{r-1} > 0$ and $\sum_{i=1}^{r-1} L_j = n - d$,

(c) $m-k'-d$ horizontal steps and $n-d$ vertical steps form s closed cycles. Let $(1, u_1, u_2, \ldots, u_{s-1}), u_r$ be the indices of the first AR in each cycle (with left ends on $Y = X - k'$). Furthermore, $l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_{s-1}}, l_r$ are the lengths of the corresponding AR where ($l_1 \geq k - k'$ for $i = 1, u_1, u_2, \ldots, u_{s-1}$ and $l_r < k - k'$) satisfying $\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j$, $v = 0, 1, \ldots, s-1$,

(d) The cycle at the end which is an open cycle made up of 1 AR of length l_{u_s}, satisfying $l_{u_s} < k, L_{u_s} = 0$,

(e) d_h diagonals are inserted each in any d_h out of $r-1$ ARs, including the vertices at both ends of the runs except for first $k_0 - k'$ vertices of first AR and first $k - k'$ vertices of AR u_i^{th}, for $i = 1, 2, \ldots, s-1$ and all vertices of r^{th} AR .

The conditions (a)-(d) above would be referred to as structural properties a LP ending with a last arrival event when the system ends in vacation period (SPLAEV) of IBP of the system under (k', k) policy.

Theorem 5.3.2. (The number of LPs satisfying to SPLAEV)
For non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k'+1), k(k-k' \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$ $\underset{\sim}{L}^* = (l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_{r-1})$, let $LP_2^1\left(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\right)$ denote the number of LP$_s$ satisfying the SPLAEV. Furthermore, the remaining $d - d_h$ diagonals are inserted each at any $n - d - r + 1$ vertices available along the DRs runs.
Then, for $k' \geq 1,\ k_0 \geq k'+1,\ k \geq k'+1,\ m \geq 2k - k',\ n \geq k - k' + 1,\ r \geq 2$ and $s \geq 1$

$$LP_2^1\left(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\right) = \sum_{(R_8^1, R_9^1)} \binom{n-d-r+1}{d-d_h} \tag{5.3}$$

where
$R_8^1 = \Big\{(i_1, i_2, \ldots, i_{d_h}) : 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r-1\Big\},$

$$R_9^1 = \left\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \le p_{i_v} \le l_{i_v}, v = 1, 2, \ldots, d_h \right\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin \big(1, u_1, u_2, \ldots, u_{s-1}\big), \\ k_0 - k', & \text{if } i_v = 1, \\ k - k', & \text{if } i_v \in \big(u_1, u_2, \ldots, u_{s-1}\big). \end{cases}$$

Proof. To proof this theorem, we may follow to the proof of Theorem 5.3.1. If we delete all the diagonal steps, and compress to form a skeleton path, then there must be $m - k' - d$ horizontal steps, and $(n - d)$ vertical steps. To get skeleton from $(k', 0)$ to $(m - k' - d, n - d)$ we suppose this skeleton consists of r ARs and $r - 1$ DRs of lengths l_i $(i = 1, 2, \ldots, r)$, and L_j $(j = 1, 2, \ldots, r - 1)$ respectively satisfying the structural conditions (a) to (e) SPLAEV.
Let d_h diagonals be inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, each of lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d - d_h$ diagonals can be inserted into any $d - d_h$ vertices out of $n - d - r + 1$ internal vertices available in the DR. This can be accomplished in $\binom{n-d-r+1}{d-d_h}$ ways. Now summing $\binom{n-d-r+1}{d-d_h}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ defined by R_8 and all insertion points $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ defined by R_9, we get (5.3). □

Lemma 5.3.2. Let $LP_2^1(k', k_0, k, m, n; s)$ be the number of lattice paths under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d - k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n - d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times, never crosses the line $Y = X - k'$, ends with a last arrival event during vacation period. Then, summing (5.3) over r, d_h, and $\underset{\sim}{L}^*$ we find

$$LP_2^1(k', k_0, k, m, n; s) = \sum_{(R_5, R_6, R_7^1)} LP_2^1(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*) \tag{5.4}$$

where

$$R_5 = \{r : 2 \le r \le \max(\min(m - k' - d, n - d), 2)\},$$
$$R_6 = \{d_h : \max(0, 2d - n + r) \le d_h \le \min(r, d)\},$$
$$R_7^1 = \Big\{ \underset{\sim}{L}^* : l_1 \ge \mathrm{Max}(k - k', L_1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \ge k - k', v = 0, 1, \ldots, s-1,\ 1 \le s \le r-1, \sum_{i=1}^{r} l_i = m - k' - d, \sum_{i=1}^{r-1} L_i = n - d, \Big\}.$$

5.3.1.3 Structural properties of a LP ending with a last arrival event occurring during busy period (SPLAEB)

Given non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k'+1), k(k-k' \geq 1), m, n; d, d_h$; $r(r \geq 2), s(\geq 1),\ l_1, l_2 \ldots, l_r; L_1, L_2, \ldots,\ L_{r-1}$, consider a lattice path under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m-d-k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n-d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times never crosses the line $Y = X - k'$. Such a LP satisfies the following structural properties:

(a) $m-k'-d$ horizontal steps form r runs of lengths $l_1, l_2, \ldots, l_r$, respectively, satisfying $l_1 \geq \text{Max}(k_0 - k', L_1), l_2, \ldots, l_r > 0$ and $\sum_{i=1}^{r} l_i = m-d-k'$,

(b) $n-d$ horizontal steps form r runs of lengths $L_1, L_2, \ldots, L_{r-1}$, respectively, satisfying $L_1, L_2, \ldots, L_{r-1} > 0$ and $\sum_{i=1}^{r-1} L_j = n-d$,

(c) $m-k'-d$ horizontal steps and $n-d$ vertical steps form s closed cycles and an open cycle ending at (m, n). Let $(1, u_1, u_2, \ldots, u_s)$ be the indices of the first AR in each cycle (with left ends on $Y = X - k'$). Furthermore, $l_1, l_{u_1}, l_{u_2}, \ldots, l_{u_s}$ are the lengths of the corresponding AR where $(l_i \geq k-k'$ for $i = 1, u_1, u_2, \ldots, u_{u_{s-1}}$ and $l_r < k-k')$ satisfying $\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j,\ v = 0, 1, \ldots, s-1$,

(d) The last cycle at the end which is an open cycle comprises of $r - u_s + 1$ AR of lengths $l_{u_s}, l_{u_s+1}, \ldots, l_r$ respectively, satisfying $l_{u_s} \geq \text{Max}\big(k - k^{'}, L_{u_s}\big), \sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i$,

(e) d_h diagonals are inserted each in any d_h out of r horizontal runs, including the vertices at both ends of the runs except for first k_0 vertices of first AR and first k vertices of $u_i^{th} AR,\ i = 1, 2, \ldots, s$.

The conditions (a)-(e) above would be referred to as structural properties a LP ending with a last arrival event when the system ends in busy period (SPLAEB) of IBP of the system under (k', k) policy. In this situations, we have the last arrival event which can occur while service is in phase 1 or phase 2. The following two Theorems present the number of LPs corresponding to SPLAEB under phase 1 and phase 2 of service, respectively.

Theorem 5.3.3. (The number of LPs satisfying SPLAEB when a last arrival event occurring during phase 1 of service, SPLAEB-I)
For non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k'+1), k(k-k' \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$ $\underset{\sim}{L}^* = l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1}$, let $LP_2^2\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big)$ denote the number of LP$_s$ satisfying the SPLAE. Furthermore, the remaining $d - d_h$ diagonals representing into phase 2 are inserted each at any $n - d - r + 1$ vertices available along the vertical runs.
Then, for $k' \geq 1,\ k_0 \geq k' + 1,\ k \geq k' + 1,\ m \geq 2k - k',\ n \geq k - k' + 1,\ r \geq 2$ and $s \geq 1$

$$LP_2^2\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big) = \sum_{(R_8^2, R_9)} \binom{n-d-r+1}{d-d_h} \tag{5.5}$$

where

$$R_8^2 = \Big\{(i_1, i_2, \ldots, i_{d_h}) : 1 \leq i_1 < i_2 < \ldots < i_{d_h} < r\Big\},$$

$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin \big(1, u_1, u_2, \ldots, u_s\big), \\ k_0 - k', & \text{if } i_v = 1, \\ k - k', & \text{if } i_v \in \big(u_1, u_2, \ldots, u_s\big). \end{cases}$$

Proof. To proof this theorem, we may follow to the proof of Theorem 5.3.1. Let us consider a LP from $(0,0)$ to $(m,n), m > n$. Delete all the d diagonal steps, and concatenate the steps to form a skeleton path consisting of AR and DR as described by vector $\underset{\sim}{L} = (l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1})$ and satisfying the structural conditions (a) to (e) SPLAEB.

Under this condition i.e. the last arrival has to occur in phase 1 of service, the insertion of any diagonal should not take place in last AR. Hence, $i_{d_h} < r$. Let d_h diagonals be inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, each of lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d - d_h$ diagonals can be inserted into any $d - d_h$ vertices out of $n - d - r + 1$ internal vertices available in the DR. This can be accomplished in $\binom{n-d-r+1}{d-d_h}$ ways. Now summing $\binom{n-d-r+1}{d-d_h}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ defined by R_8 and all insertion points $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ defined by R_9, we get (5.5). □

Lemma 5.3.3. Let $LP_2^2(k', k_0, k, m, n; s)$ be the number of lattice paths under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d - k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n - d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times, never crosses the line $Y = X - k'$, and ends with a last arrival event during busy period. Then, summing (5.5) over r, d_h, and L we find

$$LP_2^2(k', k_0, k, m, n; s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_2^2(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*) \tag{5.6}$$

where

$$R_5^1 = \{r : 2 \leq r \leq \max(\min(m - k' - d, n - d + 1), 2)\},$$
$$R_6^1 = \{d_h : \max(0, 2d - n + r + 1) \leq d_h \leq \min(r, d)\},$$
$$R_7^1 = \Big\{ \underset{\sim}{L}^* : l_1 \geq \mathrm{Max}(k - k', L_1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k - k', v =,$$
$$0, 1, \ldots, s - 1,\ 1 \leq s \leq r - 1, \textstyle\sum_{i=1}^{r} l_i = m - k' - d, \sum_{i=1}^{r-1} L_i = n - d \Big\}.$$

Theorem 5.3.4. (The number of LPs satisfying SPLAEB when a last arrival event occurring during phase 2 of service, SPLAEB-II)
For non-negative integers $k'(k' \geq 1), k_0(k_0 \geq k'+1), k(k-k' \geq 1), m, n; d, d_h; r(r \geq 2), s(\geq 1)$ $\underset{\sim}{L}^* = l_1, l_2, \ldots, l_r; L_1, L_2, \ldots, L_{r-1}$, let $LP_2^3\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big)$ denote the number of LP_s satisfying the SPLAE. Furthermore, the remaining $d - d_h$ diagonals representing into phase 2 are inserted each at any $n - d - r + 1$ vertices available along the vertical runs.
Then, for $k' \geq 1,\ k_0 \geq k' + 1,\ k \geq k' + 1,\ m \geq 2k - k',\ n \geq k - k' + 1,\ r \geq 2$ and $s \geq 1$

$$LP_2^3\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big) = \sum_{(R_8^3, R_9^2)} \begin{pmatrix} n - d - r + 1 \\ d - d_h \end{pmatrix} \tag{5.7}$$

where

$$R_8^3 = \Big\{ (i_1, i_2, \ldots, i_{d_h}) : 1 \leq i_1 < i_2 < \ldots < i_{d_h}, i_{d_h} = r \Big\},$$
$$R_9^2 = \Big\{ p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h - 1,$$
$$p_{i_v} \neq l_{i_v}, v = d_h \Big\}$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin \big(1, u_1, u_2, \ldots, u_s\big), \\ k_0 - k', & \text{if } i_v = 1, \\ k - k', & \text{if } i_v \in \big(u_1, u_2, \ldots, u_s\big). \end{cases}$$

Proof. To proof this theorem, we may follow to the proof of Theorem 5.3.1. Note that the insertion of a diagonal out of d_h should take place in last AR but not at its right end vertex. The rest of the proof is similar to that of Theorem 5.3.3. □

Lemma 5.3.4. Let $LP_2^3(k', k_0, k, m, n; s)$ be the number of lattice paths under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d - k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n - d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times, never crosses the line $Y = X - k'$, and ends with a last arrival event happened in phase 2 of service. Then, summing (5.7) over r, d_h, and $\underset{\sim}{L}^*$ we find

$$LP_2^3(k', k_0, k, m, n; s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_2^3(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*) \tag{5.8}$$

where

$$R_5^1 = \{r : 2 \le r \le \max(\min(m - k' - d, n - d + 1), 2)\},$$
$$R_6^1 = \{d_h : \max(0, 2d - n + r + 1) \le d_h \le \min(r, d)\},$$
$$R_7^1 = \Big\{\underset{\sim}{L}^* : l_1 \ge \text{Max}(k-k', L_1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \ge k-k', v =,$$
$$0, 1, \ldots, s-1,\ 1 \le s \le r-1, \textstyle\sum_{i=1}^{r} l_i = m - k' - d, \sum_{i=1}^{r-1} L_i = n - d\Big\}.$$

5.3.1.4 Structural properties of a LP ending with a last diagonal event (SPLDgE)

The structural properties of the incomplete busy period of $M/C_2/1$ system under (k', k) control policy when the system ending on a diagonal preceded by a departure (an arrival) is same with SPLDE (SPLAEB).

Theorem 5.3.5. (The number of LPs satisfying SPLDgE when the LP ends with a diagonal preceded by a departure)

For non-negative integers $k', k, m, n; d, d_h;\ r(r \ge 1), s(\ge 1),\ l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_r$, let $LP_3^1\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}\big)$ denote the number of LP_s satisfying the SPLDE. Furthermore, the remaining $d - d_h - 1$ diagonals are inserted each at any $n - d - r$ vertices available along the vertical runs.

Then, for $k' \geq 1,\ k \geq k'+1,\ m \geq 2k-k',\ n \geq k-k'+1,\ r \geq 2$ and $s \geq 1$

$$LP_3^1\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}\big) = \sum_{(R_8, R_9)} \begin{pmatrix} n-d-r \\ d-d_h-1 \end{pmatrix} \tag{5.9}$$

where

$$R_8 = \Big\{(i_1, i_2, \ldots, i_{d_h}) : 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r\Big\},$$

$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin \big(1, u_1, u_2, \ldots, u_s\big), \\ k_0 - k', & \text{if } i_v = 1, \\ k - k', & \text{if } i_v \in \big(u_1, u_2, \ldots, u_s\big). \end{cases}$$

Proof. To proof this theorem, we may follow to the proof of Theorem 5.3.1. If we delete all the diagonal steps, and compress to form a skeleton path, then there must be $m-k'-d$ horizontal steps, and $(n-d)$ vertical steps. To get skeleton from $(k', 0)$ to $(m-k'-d, n-d)$ we suppose this skeleton consists of r horizontal runs and r vertical runs of lengths $l_i(i = 1, 2, \ldots, r)$, and $L_j(j = 1, 2, \ldots, r)$ respectively. One unique path will be produced by this scenario. For the purpose of insertion, suppose d_h diagonals are inserted into runs numbered $i_1, i_2, \ldots, i_{d_h}$, respectively with lengths of $l_{i_1}, l_{i_2}, \ldots, l_{i_{d_h}}$ at distances $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$ from the extreme left end points. The remaining $d-d_h$ diagonals will be inserted into any $d-d_h$ vertices out of $n-d-r$. To complete the proof, we note that out of $d-d_h$ diagonals need to be inserted, one diagonal should be inserted at the end of lattice path. Hence, the number to do this is $\begin{pmatrix} n-d-r \\ d-d_h-1 \end{pmatrix}$.

Now summing $\begin{pmatrix} n-d-r \\ d-d_h-1 \end{pmatrix}$ over all possible d_h-tuples, $i_1, i_2, \ldots, i_{d_h}$ and $p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}$, we get (5.9). □

Lemma 5.3.5. Let $LP_3^1(k', k_0, k, m, n; d, s)$ be the number of lattice paths under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m-d-k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n-d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times, never crosses the line $Y = X - k'$, and ends with a diagonal preceded by a departure. Then, summing (5.9) over r, d_h, and L we find

$$LP_3^1(k', k_0, k, m, n; d, s) = \sum_{(R_5, R_6, R_7)} LP_3^1(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}) \qquad (5.10)$$

where

$$R_5 = \{r : 2 \le r \le \max(\min(m - k' - d, n - d + 1), 2)\}$$

$$R_6 = \{d_h : \max(0, 2d - n + r - 1) \le d_h \le \min(r, d)\}$$

$$R_7 = \Big\{ \underset{\sim}{L} : l_1 \ge \mathrm{Max}(k - k', L_1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \ge k - k', v = 0, 1, \ldots, s-1,\ 1 \le s \le r-1, \sum_{i=1}^{r} l_i = m - k' - d, \sum_{i=1}^{r} L_i = n - d, \Big\}$$

Theorem 5.3.6. (The number of LPs satisfying SPLDgE when the LP ends with a diagonal preceded by an arrival)
For non-negative integers $k', k, m, n; d, d_h; r(r \geq 1), s(\geq 1),\ l_1, l_2, \ldots, l_r, L_1, L_2, \ldots, L_{r-1}$, let $LP_3^2\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big)$ denote the number of LP$_s$ satisfying the SPLAEB. Furthermore, the remaining $d - d_h - 1$ diagonals are inserted each at any $n - d - r$ vertices available along the vertical runs.

Then, for $k' \geq 1,\ k \geq k' + 1,\ m \geq 2k - k',\ n \geq k - k' + 1,\ r \geq 2$ and $s \geq 1$

$$LP_3^2\big(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*\big) = \sum_{(R_8^2, R_9)} \begin{pmatrix} n - d - r \\ d - d_h - 1 \end{pmatrix} \tag{5.11}$$

where

$$R_8^2 = \Big\{(i_1, i_2, \ldots, i_{d_h}) : 1 \leq i_1 < i_2 < \ldots < i_{d_h} < r\Big\},$$

$$R_9 = \Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big) \colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$$

$$\triangle = \begin{cases} 0, & \text{if } i_v \notin \big(1, u_1, u_2, \ldots, u_s\big), \\ k_0 - k', & \text{if } i_v = 1, \\ k - k', & \text{if } i_v \in \big(u_1, u_2, \ldots, u_s\big). \end{cases}$$

Proof. To proof this theorem, we may follow to the proof of Theorem 5.3.1. The proof is similar to that of Theorem 5.3.3. □

Lemma 5.3.6. Let $LP_3^2(k', k_0, k, m, n; d, s)$ be the number of lattice paths under (k', k) control policy from $(k_0, 0)$ to $(m, n), m > n$ comprising of $m - d - k'$ horizontal steps including those from $(k', 0)$ to $(k_0, 0)$, $n - d$ vertical steps and d diagonals such that the LP touches $Y = X - k'$ exactly s times, never crosses the line $Y = X - k'$, and ends with a diagonal preceded by an arrival. Then, summing (5.11) over r, d_h, and $\underset{\sim}{L}^*$ we find

$$LP_3^2(k', k_0, k, m, n; d, s) = \sum_{(R_5^1, R_6^1, R_7^1)} LP_3^2(k', k_0, k, m, n; d, d_h; r, s, \underset{\sim}{L}^*) \tag{5.12}$$

where

$$R_5^1 = \{r : 2 \leq r \leq \max(\min(m - k' - d, n - d + 1), 2)\}$$

$$R_6^1 = \{d_h : \max(0, 2d - n + r - 1) \leq d_h \leq \min(r, d)\}$$
$$R_7^1 = \Big\{ \underset{\sim}{L}^* : l_1 \geq \mathrm{Max}(k - k', L_1), \textstyle\sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k - k', v = 0, 1, \ldots, s-1,\ 1 \leq s \leq r-1, \sum_{i=1}^{r} l_i = m - k' - d, \sum_{i=1}^{r-1} L_i = n - d \Big\}.$$

Note, for the case $r \geq 2$ and $d = d_h = 0$, we get

$$LP_1(k_0, k, m, n; 0, s) = LP_2^1(k_0, k, m, n; 0, s) = LP_2^2(k_0, k, m, n; 0, s) = LP_2^3(k_0, k, m, n; 0, s)$$
$$= LP_3^1(k_0, k, m, n; 0, s) = LP_3^2(k_0, k, m, n; 0, s))$$
$$= \begin{pmatrix} m + n - s(k - k') - k_0 \\ m \end{pmatrix} - \begin{pmatrix} m + n - s(k - k') - k_0 \\ m - k' \end{pmatrix} \tag{5.13}$$

5.3.2 Probability density function of IBP of $M/C_2/1$ system under (k', k) control policy

Let $f^i_{k_0;k',k}(t)$, $i = 1, 2, ..., 8$ denote the probability density function for IBP of $M/C_2/1$ system under $(k^{'}, k)$ control policy where the index i corresponds to the scenario as listed in Table 5.3. Note that k_0 denotes the initial number of customers in the system when the system opens. The corresponding density functions are presented in Theorem 5.3.7 - Theorem 5.3.15, and the key results used in the corresponding proof from previous section are presented in the last column.

In the following theorems we repeatedly use the following notation. Given a IBP of length t we partition t into three portions τ, t_1 and t_2 denoting respectively the total periods of vacation, phase 1 service and phase 2 service such that $\tau = t_1 + t_2$ and $t_2 \geq 0$. Excluding the number of transitions from phase 1 service to phase 2 service, the number of transitions during IBP of length t is partitioned into N_0, N_1 and N_2 denoting the number of transitions during τ, t_1 and t_2 respectively.

Table 5.3: The scenarios of LP representation of IBP of $M/C_2/1$

Index i	**Scenario at time** t	**Theorem from Section 5.3.1**
1	No phase 2 service and system of vacation	-
2	No phase 2 service and system of busy	-
3	The corresponding LP satisfies SPLDE with last departure event	5.3.1
4	The corresponding LP satisfies SPLAEV with last arrival event under vacation	5.3.2
5	The corresponding LP satisfies SPLAEB with last arrival event under phase 1 of service	5.3.3
6	The corresponding LP satisfies SPLAEB with last arrival event under phase 2 of service	5.3.4
7	The corresponding LP satisfies SPLDE with last diagonal event preceded by a departure	5.3.5
8	The corresponding LP satisfies SPLAEB with last diagonal event preceded by an arrival	5.3.6

The parameter sets used in defining pdf is presented in Table 5.4.

Table 5.4: The parameter sets used in defining pdf

Notation	Description
R_1	$= \{m : k_0 + k - k^{'} \leq m < \infty\}.$
R_1^1	$= \{k^* : 0 \leq k^* \leq k - k^{'} - 1\}.$
R_1^2	$= \{m : k_0 + k - k^{'} + 1 \leq m < \infty\}.$
R_1^3	$= \{m : k_0 + k - k^{'} + 2 \leq m < \infty\}.$
R_1^4	$= \{m : k_0 + k - k^{'} + 3 \leq m < \infty\}.$
R_2	$= \{n : k_0 \leq n < m - k^*\}.$
R_2^1	$= \{n : k_0 + 1 \leq n \leq m - 1\}.$
R_2^2	$= \{n : k_0 + 2 \leq n \leq m - 1\}.$
R_2^3	$= \{n : k_0 + 3 \leq n \leq m - 1\}.$
R_3	$= \left\{s : 1 \leq s < \left\lfloor \frac{n-k_0}{k} + 1 \right\rfloor \right\}.$
R_4	$= \left\{d : 1 \leq d \leq \min\left(\lfloor \frac{n}{2} \rfloor, m - k_0\right)\right\}$ where $[x]$ denotes the largest integer contained in x.
R_5	$= \{r : 2 \leq r \leq \max(\min(m - k^{'} - d, n - d), 2)\}.$
R_5^1	$= \{r : 2 \leq r \leq \max(\min(m - k^{'} - d, n - d + 1), 2)\}.$
R_6	$= \{d_h : \max(0, 2d - n + r) \leq d_h \leq \min(r, d)\}.$
R_6^1	$= \{d_h : \max(0, 2d - n + r - 1) \leq d_h \leq \min(r, d)\}.$
R_7	$= \left\{\underset{\sim}{L} : l_1 \geq \text{Max}(k_0, L_1 + 1), \sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k\right.$ $v = 0, 1, \ldots, s-1,\ l_{u_s} \geq \max\left(k, L_{u_s}\right), \sum_{i=u_s}^{r-1} l_i > \sum_{i=u_s}^{r-1} L_i,$ $\left.\sum_{i=1}^{r} l_i = m - d - k', \sum_{i=1}^{r} L_i = n - d\right\}.$
R_7^1	$= \left\{\underset{\sim}{L}^* : l_1 \geq \text{Max}(k - k^{'}, L_1), \sum_{j=u_v}^{u_{v+1}-1} l_j = \sum_{j=u_v}^{u_{v+1}-1} L_j, l_{u_v} \geq k - k^{'}\right.$ $v = 0, 1, \ldots, s-1, 1 \leq s \leq r - 1, \sum_{i=u_s}^{r} l_i > \sum_{i=u_s}^{r} L_i,$ $\left.\sum_{i=1}^{r} l_i = m - k^{'} - d, \sum_{i=1}^{r} L_i = n - d\right\}.$
R_8	$= \left\{\left(i_1, i_2, \ldots, i_{d_h}\right): 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r\right\}.$
R_8^1	$= \left\{\left(i_1, i_2, \ldots, i_{d_h}\right): 1 \leq i_1 < i_2 < \ldots < i_{d_h} \leq r - 1\right\}.$
R_8^2	$= \left\{\left(i_1, i_2, \ldots, i_{d_h}\right): 1 \leq i_1 < i_2 < \ldots < i_{d_h} < r\right\}.$
R_8^3	$= \left\{\left(i_1, i_2, \ldots, i_{d_h}\right): 1 \leq i_1 < i_2 < \ldots < i_{d_h}, i_{d_h} = r\right\}.$

Notation		**Description**
R_9	$=$	$\Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big)\colon \triangle \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$
		$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_{s-1}), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_{s-1}). \end{cases}$
R_9^1	$=$	$\Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big)\colon \triangle_1 \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h\Big\},$
		$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s - 1), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s - 1). \end{cases}$
R_9^2	$=$	$\Big\{p_i = \big(p_{i_1}, p_{i_2}, \ldots, p_{i_{d_h}}\big)\colon \triangle_1 \leq p_{i_v} \leq l_{i_v}, v = 1, 2, \ldots, d_h - 1,$ $p_{i_v} \neq l_{i_v}, v = d_h\Big\},$
		$\triangle_1 = \begin{cases} 0, & \text{if } i_v \notin (1, u_1, u_2, \ldots, u_s - 1), \\ k_0, & \text{if } i_v = 1, \\ k, & \text{if } i_v \in (u_1, u_2, \ldots, u_s - 1). \end{cases}$
α_1	:	the probability that a customer will move from phase 1 to the phase 2.
β	:	the probability that a customer will leave the system completely after completing phase 1 of service.
λ	:	the average rate of customers entering the queueing system.
μ_1	:	the average rate of serving customers in phase 1.
μ_2	:	the average rate of serving customers in phase 2.

Theorem 5.3.7. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy such that no customer receives phase 2 service and ends in vacation period)

Let $f^1_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ system starting initially with k_0 customers at time t under (k', k) policy without encountering phase 2 service and ending in vacation period. Then we have

$$
\begin{aligned}
f^1_{k_0;(k',k)}(t) =& e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_1^1,R_2,R_3)} \frac{k_0+(s-1)(k-k')}{2n-k_0-(s-1)(k-k')} \\
&\times \begin{pmatrix} 2n-k_0-(s-1)(k-k') \\ n \end{pmatrix} \\
&\times \frac{\lambda^{m-k_0}\beta^n}{\Gamma(k^*+(s-1)(k-k')+1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \\
&\times \frac{\Gamma(k^*+(s-1)(k-k')+i+1)}{\Gamma(m+n-k_0+i+1)}
\end{aligned} \tag{5.14}
$$

where $R_1, R_1^1, R_2, R_3, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The above equation corresponds to the case when all customers depart after having phase 1 service and system ends in vacation period i.e. in the corresponding LP, $d = 0$ and after the s^{th} (last) touch occurs, there are $k^*(< k-k')$ horizontal steps up to (m, n) where $m > n$ and $m = n + k' + k^*$, $k^*(< k-k')$, $k^* = 0, 1, 2, \ldots, (k-k'-1)$.

To prove the identity (5.14), first, we need to count the number oflattice paths under scenario 1 of Table 5.3. For this, let $AB_1 \ldots B_sB$ be a lattice path (Figure 5.3) stipulated in structural properties for last arrival event (SPLAEV) such that the path from $(k_0, 0)$ to (m, n) touches s times the line $Y = X - k'$ at the points $B_1, \ldots, B_s$. After s^{th}(last) touch, there are $k^*(< k-k')$ horizontal steps up to (m, n).

First, ignore the last segment B_sB. Then, we have the set of paths from $(k_0, 0)$ to (n, n) having s touches the line $Y = X-k'$ at the points $B_1, \ldots, B_s$. A 1:1 mapping is constructed by removing the first k horizontal steps from $B_1B_2, \ldots, B_{s-1}B_s$. We obtain a 1:1 set of paths from $(k_0, 0)$ to $(n-(s-1)(k-k'), n)$ which do not touch the line $Y = X - k' + (s-1)(k-k')$. Its number is given by

$$
\frac{k_0+(s-1)(k-k')}{2n-k_0-(s-1)(k-k')} \begin{pmatrix} 2n-k_0-(s-1)(k-k') \\ n \end{pmatrix}
$$

Next, let $t = \tau + t_1$ be the total time spent in the system, where τ is the total

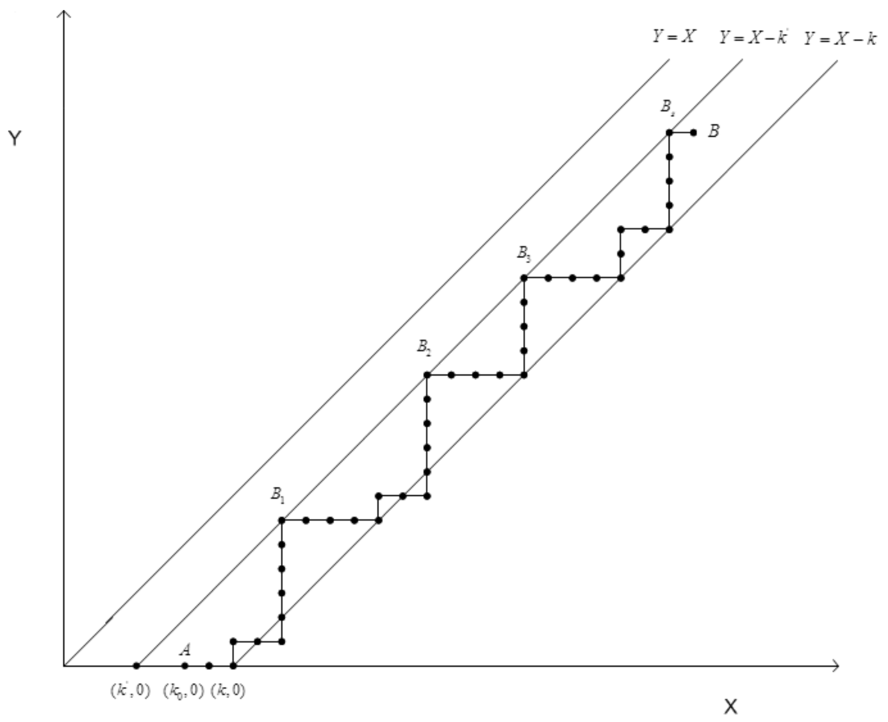


Figure 5.3: A lattice path stipulated by SPLAEV

time in vacation period and t_1 is the total time spent in phase 1 of service.

The number of arrivals is $m - k_0$. This consists of $(s-1)(k-k') + k^*$ arrivals during the vacation periods and $m - (s-1)(k-k') - k^* - k_0$ arrivals during the busy periods. The number of departures is n. Therefore total number of transition is $m + n - k_0$. Call this as N_0.

Let $\tau = \tau_1 + \tau_2 + \ldots + \tau_s$ be the total time system is in vacation where τ_i is the vacation time system in $u_i, i = 1, 2, \ldots, s$. The transitions in each τ_i for $i = 1, 2, \ldots, s-1$ consist of $k - k'$ arrivals while the transition in τ_s consist of k^* arrivals. The number of arrivals is $(s-1)(k-k') + k^*$. The probability density function of τ for given parameter is given by

$$f_{s,k,k',k^*}(\tau) = \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}\lambda^{k-k'}\tau_y^{k-k'-1}}{\Gamma(k-k')} \frac{e^{-\lambda\tau_y}\lambda^{k^*}\tau_s^{k^*-1}}{\Gamma(k^*)} d\tau_s \ldots d\tau_1 =$$

$$\frac{e^{-\lambda\tau}\lambda^{(s-1)(k-k')+k^*}}{(\Gamma(k-k'))^{s-1}\Gamma(k^*)} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \tau_y^{k-k'-1}\tau_s^{k^*-1} d\tau_s \ldots d\tau_1$$

which is equal to

$$f_{s,k,k',k^*}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{k^*+(s-1)(k-k')}}{\Gamma(k^* + (s-1)(k-k') + 1)}.$$

The derivation of this pdf is given in Appendix B.6.

The total number of transition in t_1, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as explained below:

Number of arrivals: $m-(s-1)(k-k')-k^*-k_0$ since $(s-1)(k-k')+k^*$ is the number of arrivals while system is idle period.

Number of departures after phase 1: n,

Note that the total number of transition during the period t_1 is

$$N_1 = m + n - (s-1)(k-k') - k^* - k_0.$$

The probability density function of $t_1 = t - \tau$ is $N_1 - Erlang$ with parameter $(\lambda + \mu_1)$ given by

$$f_{N_1}(t-\tau) = \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{N_1}(t-\tau)^{N_1-1}}{\Gamma(N_1)}$$

The probability density function of this IBP is

$$\begin{aligned} f^1_{k_0;(k',k)}(t) = & \sum_{(R_1,R_1^1,R_2,R_3)} \frac{k_0+(s-1)(k-k')}{2n-k_0-(s-1)(k-k')} \begin{pmatrix} 2n-k_0-(s-1)(k-k') \\ n \end{pmatrix} \\ & \times \int_{\tau=0}^{t} \frac{e^{-\lambda\tau}(\lambda\tau)^{k^*+(s-1)(k-k')}}{\Gamma((s-1)(k-k')+k^*+1)} \\ & \times \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{m+n-(s-1)(k-k')-k^*-k_0}(t-\tau)^{m+n-(s-1)(k-k')-k^*-k_0-1}}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\ & \times \left(\frac{\lambda}{\lambda+\mu_1}\right)^{m-(s-1)(k-k')-k^*-k_0} \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n} d\tau \end{aligned}$$

The integral part is calculated and given in Appendix B.7.

Hence, we establish the density function for this IBP at time t, i.e.

$$\begin{aligned} f^1_{k_0;k',k}(t) = & e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_1^1,R_2,R_3)} \frac{k_0+(s-1)(k-k')}{2n-k_0-(s-1)(k-k')} \begin{pmatrix} 2n-k_0-(s-1)(k-k') \\ n \end{pmatrix} \\ & \times \frac{\lambda^{m-k_0}\beta^n}{\Gamma(k^*+(s-1)(k-k')+1)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma(k^*+(s-1)(k-k')+i+1)}{\Gamma(m+n-k_0+i+1)} \end{aligned}$$

□

Theorem 5.3.8. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy with no customer receiving phase 2 service and the system ends in busy period)
Let $f^2_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (k', k) policy where system starts initially with k_0 customers, has no phase 2 service and ends in busy period. Then we have

$$\begin{aligned} f^2_{k_0;k',k}(t) =& e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_2,R_3)} \left\{ \left(\begin{array}{c} m+n-s(k-k')-k_0 \\ n \end{array} \right) - \right. \\ &\times \left. \left(\begin{array}{c} m+n-s(k-k')-k_0 \\ m-k' \end{array} \right) \right\} \lambda^{m-k_0} \frac{\beta^n}{\Gamma(s(k-k')+1)} \\ &\times \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(m+n-k_0+i+1)} \end{aligned} \tag{5.15}$$

where $R_1, R_2, R_3, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The above equation to the case when all customers depart after having phase 1 service and so in the corresponding LP we have $d = 0$. To prove the identity (5.15), let $t = \tau + t_1$ be the total time spent in the system, where τ is the total time in vacation state, t_1 is the total time spent in phase 1 of service.

The number of arrivals is $m - k_0$. This consists of $s(k - k')$ arrivals during the vacation periods and $m - s(k - k') - k_0$ arrivals during the busy periods. The number of departures is n. Therefore total number of transition is $m + n - k_0$. Call this as N_0.

Let $\tau = \tau_1 + \tau_2 + \ldots + \tau_s$, the total time system is in vacation where τ_i is the vacation time system in $u_i, i = 1, 2, \ldots, s$. The transitions in τ_i consist of only arrivals. The number of arrivals is $s(k - k')$. The probability of τ for given parameter is given by

$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y} \lambda^{k-k'} \tau_y^{k-k'-1}}{\Gamma(k-k')} d\tau_y = \frac{e^{-\lambda\tau_y} \lambda^{s(k-k')}}{(\Gamma(k-k'))^s} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s} \tau_y^{k-k'-1} d\tau_y$$

which is equal to

$$f_{s(k-k')}(\tau) = \frac{e^{-\lambda\tau} (\lambda\tau)^{s(k-k')}}{\Gamma(s(k-k')+1)}$$

The derivation of this pdf is given in Appendix B.8.

The total number of transition in t_1, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as explained below:

Number of arrivals: $m - k_0 - s(k - k')$ since $s(k - k')$ is the number of while system is idle period.

Number of departures after phase 1: n,

The total number of transition during the phase 1 service time t_1 is

$$N_1 = m + n - k_0 - s(k - k'),$$

The probability density function of $t_1 = t - \tau$ is $N_1 - Erlang$ with parameter $(\lambda + \mu_1)$ given by

$$f_{N_1}(t - \tau) = \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda + \mu_1)^{N_1}(t - \tau)^{N_1-1}}{\Gamma(N_1)}$$

The probability density function of $t_1 = t - \tau$ is

$$\int_{\tau=0}^{t} \frac{e^{-\lambda\tau}\lambda^{s(k-k')}\tau^{s(k-k')}}{\Gamma(s(k - k') + 1)} \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda + \mu_1)^{m+n-k_0-s(k-k')}(t - \tau)^{m+n-k_0-s(k-k')-1}}{\Gamma(m + n - k_0 - s(k - k'))}$$
$$\times \left(\frac{\lambda}{\lambda + \mu_1}\right)^{m-k_0-s(k-k')} \left(\frac{\beta\mu_1}{\lambda + \mu_1}\right)^{n} d\tau$$

Let $\upsilon = \frac{\tau}{t}$ then the above expression becomes

$$e^{-(\lambda+\mu_1)t} \frac{\lambda^{m-k_0}}{\Gamma(s(k - k') + 1)} \frac{\beta^n}{\Gamma(m + n - k_0 - s(k - k'))} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i}$$
$$\times B(s(k - k') + i + 1, m + n - k_0 - s(k - k')). \tag{5.16}$$

For counting the lattice paths, let $AB_1 \dots B_sB$ be a lattice path (Figure 5.4) stipulated in Theorem 5.4.4 in which the path from $(k_0, 0)$ to $(m, n), m > n$, touches s times the line $Y = X - k'$ the points $B_1, \dots, B_s$.

A 1:1 mapping is constructed by removing the first $k - k'$ horizontal steps from $B_1B_2, \dots, B_{s-1}B_s$, and getting the set of paths from $(k_0, 0)$ to $(m - s(k - k'), n)$ which do not touch the line $Y = X - (k' - s(k - k'))$. Its number is given by

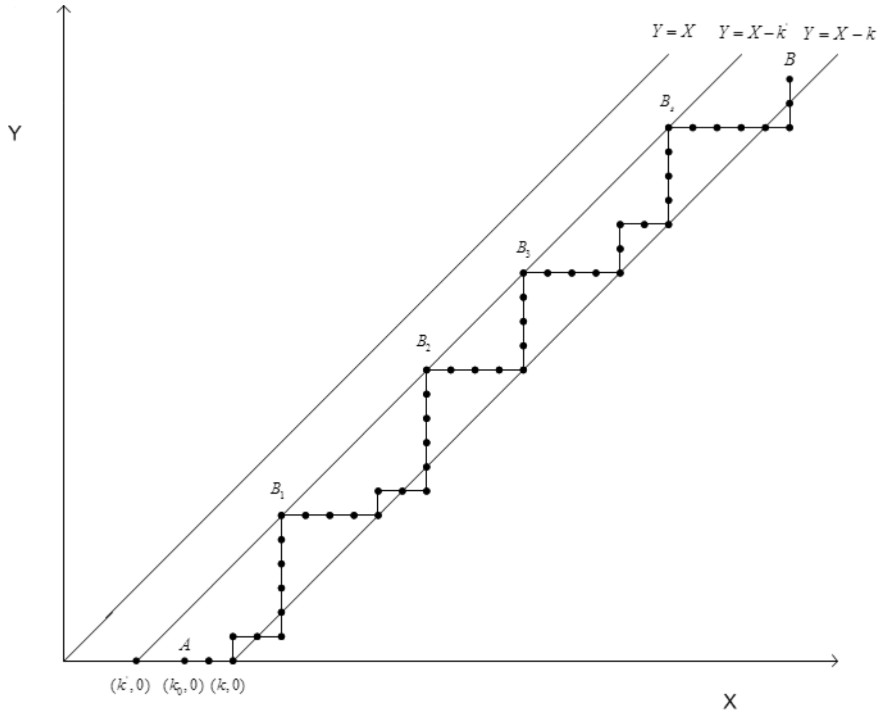


Figure 5.4: A lattice path stipulated by SPLAEB

$$\binom{m+n-s(k-k')-k_0}{n} - \binom{m+n-s(k-k')-k_0}{m-k'}$$

Multiplying this number and (5.16), we establish the density function for this IBP at time t, i.e.

$$\begin{aligned} f^2_{k_0;k',k}(t) = & \sum_{(R_1,R_2,R_3)} \binom{m+n-s(k-k')-k_0}{n} - \binom{m+n-s(k-k')-k_0}{m-k'} \\ & \times e^{-(\lambda+\mu_1)t} \frac{\lambda^{m-k_0}}{\Gamma(s(k-k')+1)} \frac{\beta^n}{\Gamma(m+n-k_0-s(k-k'))} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \\ & \times \frac{\Gamma(s(k-k')+i+1)\Gamma(m+n-k_0-s(k-k'))}{\Gamma(m+n-k_0+s+i+1)} \end{aligned}$$

Simplify the above equation, yields

$$\begin{aligned} f^2_{k_0;k',k}(t) =& e^{-(\lambda+\mu_1)t} \sum_{(R_1,R_2,R_3)} \left\{ \begin{pmatrix} m+n-s(k-k')-k_0 \\ n \end{pmatrix} - \right. \\ & \times \left. \begin{pmatrix} m+n-s(k-k')-k_0 \\ m-k' \end{pmatrix} \right\} \lambda^{m-k_0} \frac{\beta^n}{\Gamma(s(k-k')+1)} \\ & \times \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(m+n-k_0+i+1)} \end{aligned}$$

□

Theorem 5.3.9. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy ending with last departure event)
Let $f^3_{k_0;k',k}(t)$ denote the probability density function of IBP busy period of $M/C_2/1$ under (k', k) policy given that system starts initially with k_0 customers and ends with last departure event. Then we have

$$\begin{aligned} f^3_{k_0;k',k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^2,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d} \\ & \times \lambda^{m-d-k_0} \mu_1^{n-d} \mu_2^d \sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \\ & \times \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j)} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(s(k-k')+1)} \\ & \times \frac{\Gamma\big(m+n-2d-k_0-s(k-k')+j-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}, \end{aligned} \tag{5.17}$$

where $R_1^2, R_2^1, R_3, R_4, R_5, R_6, R_7, R_8, R_9, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The RHS of (5.17) corresponds to the the scenario 3 of Table 5.3 , that is to the density of the system IBP such that the system still busy while ending with a departure after phase 1 of service as well as phase 2 of service.
The term for which departure occurs can be derived as follow. Let $t = \tau + t_1 + t_2$ be the total time spent in the system, where τ is the total vacation time, t_1 is the total time spent in phase 1 of service and $t_2 = t - \tau - t_1$ is the total time spent in phase 2 of service.

For fixed $\underset{\sim}{L}$ of a LP of a IBP of length t, total number of transitions when system still busy is given by $m+n-d-k_0$ as described below:

Number of arrivals during the vacation period: $s(k-k')$.

Number of arrivals during the busy period: $m-s(k-k')-d-k_0$.

Number of departures after phase 1: $n-2d$.

Number entries into phase 2: d.

Number of departures after phase 2: d.

The number of transition in τ, the time system is in vacation period consists of only arrivals. The number of arrivals is $s(k-k')$. The probability of τ is given by

$$f_{s(k-k')}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{s(k-k')}}{\Gamma(s(k-k')+1)}.$$

The total number of transition in t_1, the time spent in phase 1 of service, consists of arrivals and departures while customers are in phase 1 of service as well as entries into phase 2 of service as explained below:

Number of arrivals: $m-d-k_0-s(k-k')-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$ since $s(k-k')$ is the number of arrivals while system is vacation period and $\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$ is the number of arrivals while customers are in phase 2 of service.

Number of entries into phase 2: d.

Number of departures after phase 1: $n-2d$.

The total number of transition during phase 1 service time t_1 is

$$N_1 = m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big), \tag{5.18}$$

The probability density function of phase 1 service time t_1 is $N_1-Erlang$ with parameter $(\lambda+\mu_1)$ given by

$$f_{N_1}(t_1) = \frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}t_1^{N_1-1}}{\Gamma(N_1)}.$$

The total number of transition in phase 2 service time $t_2 = t-\tau-t_1$ consists of arrivals and departures while customers are in phase 2 of service as explained below:

Number of arrivals: $\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$.

Number of departures: d.

Therefore N_2, total number of transition during $t_2 = t - \tau - t_1$ will be

$$N_2 = \sum_{v=1}^{d_h} \big(l_{i_v} - p_{i_v}\big) + d.$$

The probability density function of t_2 is $N_2 - Erlang$ with parameter $(\lambda + \mu_2)$ given by

$$f_{N_2}(t_2) = \frac{e^{-(\lambda+\mu_2)t_2}(\lambda+\mu_2)^{N_2} t_2^{N_2-1}}{\Gamma(N_2)}$$

Using Theorem 5.4.1, the probability density function of IBP of $M/C_2/1$ under (k', k) policy where system starts initially with k_0 customers and ends with departure is given by

$$\begin{aligned}
f^3_{k_0;k',k}(t) = & \sum_{(R_1^2,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \\
& \times \int_0^t \int_0^{t-\tau} f_{s(k-k')}(\tau) f_{N_1}(t_1) \left(\frac{\lambda}{\lambda+\mu_1}\right)^{N_1-n+d} \left(\frac{\alpha_1\mu_1}{\lambda+\mu_1}\right)^{d} \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n-2d} \\
& \times f_{N_2}(t-\tau-t_1) \left(\frac{\lambda}{\lambda+\mu_2}\right)^{N_2-d} \left(\frac{\mu_2}{\lambda+\mu_2}\right)^{d} dt_1 d\tau \\
= & \sum_{(R_1^2,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \\
& \times \int_0^t \int_0^{t-\tau} \frac{e^{-\lambda\tau}\lambda^{s(k-k')}\tau^{s(k-k')}}{\Gamma(s(k-k')+1)} \frac{e^{-(\lambda+\mu_1)t}(\lambda+\mu_1)^{N_1} t_1^{N_1-1}}{\Gamma(N_1)} \left(\frac{\lambda}{\lambda+\mu_1}\right)^{N_1-n+d} \\
& \left(\frac{\alpha_1\mu_1}{\lambda+\mu_1}\right)^{d} \left(\frac{\beta+\mu_1}{\lambda+\mu_1}\right)^{n-2d} \\
& \times \frac{e^{-(\lambda+\mu_2)}(\lambda+\mu_2)^{N_2}(t-\tau-t_1)^{N_2-1}}{\Gamma(N_2)} \left(\frac{\lambda}{\lambda+\mu_2}\right)^{N_2-d} \left(\frac{\mu_2}{\lambda+\mu_2}\right)^{d} dt_1 d\tau \\
= & \sum_{(R_1^2,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \\
& \times \alpha^d \beta^{n-2d} \lambda^{m-d-k_0} \mu_1^{n-d} \mu_2^d \frac{1}{\Gamma(s(k-k')+1)\Gamma(N_1)\Gamma(N_2)} e^{-(\lambda+\mu_2)t} \\
& \times \int_0^t \int_0^{t-\tau} e^{\mu_2\tau} e^{(\mu_2-\mu_1)t_1} t_1^{N_1-1} \tau^{s(k-k')} (t-\tau-t_1)^{N_2-1} dt_1 d\tau.
\end{aligned} \tag{5.19}$$

The derivation of double integral is given in Appendix B.9.

Substituting (B.9) in Appendix B.9 in (5.19) for values of N_1 and N_2, we get

$$
\begin{aligned}
f^{3}_{k_0;(k',k)}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R^2_1,R^1_2,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d} \\
&\times \lambda^{m-d-k_0} \mu_1^{n-d} \mu_2^d \sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \\
&\times \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j)} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(s(k-k')+1)} \\
&\times \frac{\Gamma\big(m+n-2d-k_0-s(k-k')+j-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)},
\end{aligned}
$$

Hence the proof is complete. □

Theorem 5.3.10. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy such that last arrival event happened in vacation period)
Let $f^4_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (k', k) policy given that system starts initially with k_0 customers and the last arrival event ends in vacation period (corresponding to a case under the scenario 4 of Table 5.3). Then we have

$$
\begin{aligned}
f^4_{k_0;k',k}(t) = & \sum_{(R_1^2, R_2^1, R_3, R_4, R_5, R_6, R_7^1, R_8^1, R_9^1)} \binom{n-d-r+1}{d-d_h} \alpha^d \beta^{n-2d} \\
& \times \lambda^{m-d-k_0} \mu_1^{n-d} \mu_2^d e^{-(\lambda+\mu_2)t} \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \\
& \times \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \frac{\Gamma((s-1)(k-k')+k^*+i+1)}{\Gamma((s-1)(k-k')+k^*+1)} \\
& \times \frac{\Gamma\big(m+n-2d-k_0-(s-1)(k-k')+k^*+j-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-(s-1)(k-k')+k^*-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)},
\end{aligned}
\tag{5.20}
$$

where $R_1^2, R_2^1, R_3, R_4, R_5, R_6, R_7^1, R_8^1, R_9^1, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The identity in (5.20) corresponds to a case of scenario 4 of Table 5.3 when the last arrival event ends in vacation period. We present the summary of the event and the transitions of $M/C_2/1$ system in vacation period (τ), in phase 1 service (t_1), and in phase 2 service (t_2) in Table 5.5. Following the proof of Theorem 5.4.9, we construct appropriate Erlang distribution for vacation period, (t_1) and (t_2) and present them in last row of Table 5.5. Note $N_1 = m+n-2d-k_0-(s-1)(k-k')-k^*-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})$ and $N_2 = \sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})+d$. By integrating over the product of each Erlang distribution and the corresponding number of lattice paths, we prove (5.20).

□

Table 5.5: Transitions for $M/C_2/1$ for vacation period, t_1 and t_2: The system ends with an arrival in vacation period

Event	**Vacation period**	t_1	t_2
Number of arrivals	$(s-1)(k-k')-k^*$	$m-d-k_0-(s-1)(k-k')$ $+k^*-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$	$\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$
Number of departure after phase 1	-	$n-2d$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	d
Total number of transition	$(s-1)(k-k')+k^*$	N_1	N_2
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{(s-1)(k-k')+k^*}}{\Gamma((s-1)(k-k')+k^*+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}}{\Gamma(N_1)}$ $\times t_1^{N_1-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}}{\Gamma(N_2)}$ $\times(t-\tau-t_1)^{N_2-1}$

Theorem 5.3.11. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy such that last arrival event happened in phase 1 of service)

Let $f^5_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (k', k) policy given that system starts initially with k_0 customers and the last arrival event is in phase 1 of service (corresponding to a case under the scenario 5 of Table 5.3). Then we have

$$
\begin{aligned}
f^5_{k_0;k',k}(t) =& e^{-(\lambda+\mu_2)t}\sum_{(R_1^3,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8^2,R_9)}\begin{pmatrix} n-d-r+1 \\ d-d_h\end{pmatrix}\alpha^d\beta^{n-2d} \\
&\times \lambda^{m-d-k_0}\mu_1^{n-d}\sum_{i=1}^{\infty}\frac{\mu_2^{d+i}}{i!}\sum_{j=1}^{\infty}\frac{(\mu_2-\mu_1)^j}{j!}\frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \\
&\times\frac{\Gamma(s(k-k')+i+1)}{\Gamma(s(k-k'))}\frac{\Gamma\Big(m+n-2d-k_0-s(k-k')+j-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\Big)}{\Gamma\Big(m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\Big)},
\end{aligned}
\tag{5.21}
$$

where $R_1^3, R_2^1, R_3, R_4, R_5, R_6, R_7, R_8^2, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The identity in (5.21) corresponds to a case of scenario 5 of Table 5.3 when the last arrival event is in phase 1 of service. We present the summary of the event and the transitions of $M/C_2/1$ system in vacation period (τ), in phase 1 service (t_1), and in phase 2 service (t_2) in Table 5.6. Following the proof of Theorem 5.4.9, we construct appropriate Erlang distribution for vacation period, (t_1) and (t_2) and present them in last row of Table 5.6. Note $N_1 = m + n - 2d - k_0 - s(k - k') - \sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$ and $N_2 = \sum_{v=1}^{d_h}(l_{i_v} - p_{i_v}) + d$. By integrating over the product of each Erlang distribution and the corresponding number of lattice paths, we prove 5.21.

Table 5.6: Transitions for $M/C_2/1$ for vacation period, t_1 and t_2: The system ends with an arrival in phase 1

Event	**Vacation period**	t_1	t_2
Number of arrivals	$s(k-k')$	$m - d - k_0 - (s-1)k$ $- \sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$	$\sum_{v=1}^{d_h}(l_{i_v} - p_{i_v})$
Number of departure after phase 1	-	$n - 2d$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	d
Total number of transition	$s(k-k')$	N_1	N_2
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{s(k-k')}}{\Gamma(s(k-k')+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}}{\Gamma(N_1)}$ $\times t_1^{N_1-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}}{\Gamma(N_2)}$ $\times (t-\tau-t_1)^{N_2-1}$

□

Theorem 5.3.12. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy such that a last arrival event happened in phase 2 of service)

Let $f^6_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (k', k) policy given that system starts initially with k_0 customers and the last arrival event is in phase 2 of service (corresponding to a case under the scenario 6 of Table 5.3). Then we have

$$\begin{aligned} f^6_{k_0;k',k}(t) =& e^{-(\lambda+\mu_2)t} \sum_{(R_1^4, R_2^2, R_3, R_4, R_5, R_6^1, R_7^1, R_8^3, R_9^2)} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d} \\ & \times \lambda^{m-d-k_0} \mu_1^{n-d-1} \mu_2^{d-1} \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \\ & \times \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j)} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(s(k-k')+1)} \\ & \times \frac{\Gamma\big(m+n-2d-k_0-s(k-k')+j-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}{\Gamma\big(m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}, \end{aligned} \tag{5.22}$$

where $R_1^4, R_2^2, R_3, R_4, R_5, R_6^1, R_7^1, R_8^3, R_9^2, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The identity in (5.22) corresponds to a case of scenario 6 of Table 5.3 when the last arrival event is in phase 2 of service. We present the summary of the event and the transitions of $M/C_2/1$ system in vacation period (τ), in phase 1 service (t_1), and in phase 2 service (t_2) in Table 5.7. Following the proof of Theorem 5.4.9, we construct appropriate Erlang distribution for vacation period, (t_1) and (t_2) and present them in last row of Table 5.7. Note Here $N_1 = m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$ and $N_2 = \sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)+d-1$. By integrating over the product of each Erlang distribution and the corresponding number of lattice paths, we prove 5.22.

□

Table 5.7: Transitions for $M/C_2/1$ for vacation period, t_1 and t_2: The system ends with an arrival in phase 2

Event	**Vacation period**	t_1	t_2
Number of arrivals	$s(k-k')$	$m-d-k_0-s(k-k')$ $-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$	$\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)$
Number of departure after phase 1	-	$n-2d$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	$d-1$
Total number of transition	$s(k-k')$	N_1	N_2
Erlang distribution	$\frac{e^{-\lambda\tau}(\lambda\tau)^{s(k-k')}}{\Gamma(s(k-k')+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_1}}{\Gamma(N_1)}$ $\times t_1^{N_1-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_2}}{\Gamma(N_2)}$ $\times(t-\tau-t_1)^{N_2-1}$

Theorem 5.3.13. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy ending with a diagonal preceded by a departure) Let $f^7_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (k', k) policy given that system starts initially with k_0 customers and ends with a diagonal preceded by a departure corresponding to scenario 7 of Table 5.3. Then we have

$$f^7_{k_0;k',k}(t) = \sum_{(R_1^2,R_2^3,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d+1}$$
$$\times \lambda^{m-d-k_0} \mu_1^{n-d+1} \mu_2^{d-1} e^{-(\lambda+\mu_2)t} \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!}$$
$$\times \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(s(k-k'))}$$
$$\times \frac{\Gamma\big(m+n-2d-k_0-s(k-k')+j+1-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}{\Gamma\big(m+n-2d-k_0-s(k-k')+1-\sum_{v=1}^{d_h}\big(l_{i_v}-p_{i_v}\big)\big)}, \tag{5.23}$$

where $R_1^2, R_2^3, R_3, R_4, R_5, R_6, R_7, R_8, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The identity in (5.23) corresponds to a case of scenario 7 of Table 5.3 ending with a diagonal preceded by a departure. We present the summary of the event and the transitions of $M/C_2/1$ system in vacation period (τ), in phase 1 service (t_1), and in phase 2 service (t_2) in Table 5.8. Following the proof of Theorem 5.4.9, we construct appropriate Erlang distribution for vacation period, (t_1) and (t_2) and present them in last row of Table 5.8. Note here $N_3 = m + n - 2d - k_0 - s(k - k') - \sum_{v=1}^{d_h} (l_{i_v} - p_{i_v}) + 1$ and $N_4 = \sum_{v=1}^{d_h} (l_{i_v} - p_{i_v}) + d - 1$. By integrating over the product of each Erlang distribution and the corresponding number of lattice paths, we prove 5.23.

Table 5.8: Transitions for $M/C_2/1$ for vacation period, t_1 and t_2: The system ends with diagonal preceded by a departure

Event	**Vacation period**	t_1	t_2
Number of arrivals	$s(k-k')$	$m - d - k_0 - s(k-k') - \sum_{v=1}^{d_h} (l_{i_v} - p_{i_v})$	$\sum_{v=1}^{d_h} (l_{i_v} - p_{i_v})$
Number of departure after phase 1	-	$n - 2d + 1$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	$d - 1$
Total number of transition	$s(k-k')$	N_3	N_4
Erlang distribution	$\frac{e^{-\lambda\tau}\lambda^{s(k-1)}\tau^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_3}}{\Gamma(N_3)} \times t_1^{N_3-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_4}}{\Gamma(N_4)} \times (t-\tau-t_1)^{N_4-1}$

□

Theorem 5.3.14. (Probability density function of IBP of $M/C_2/1$ system under (k', k) policy ending with a diagonal preceded by an arrival) Let $f^8_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ under (k', k) given that system starts initially with k_0 customers and ends with a diagonal preceded by an arrival corresponding to scenario 8 of Table 5.3. Then we have

$$
\begin{aligned}
f^8_{k_0;k',k}(t) = & \sum_{(R_1^2,R_2^3,R_3,R_4,R_5^1,R_6^1,R_7^1,R_8^2,R_9)} \begin{pmatrix} n-d-r+1 \\ d-d_h \end{pmatrix} \alpha^d \beta^{n-2d+1} \\
& \times \lambda^{m-d-k_0} \mu_1^{n-d+1} \mu_2^{d-1} e^{-(\lambda+\mu_2)t} \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \\
& \times \frac{t^{m+n-d-k_0+i+j}}{\Gamma(m+n-d-k_0+i+j+1)} \frac{\Gamma(s(k-k')+i+1)}{\Gamma(s(k-k'))} \\
& \times \frac{\Gamma\big(m+n-2d-k_0-s(k-k')+j+1-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)}{\Gamma\big(m+n-2d-k_0-s(k-k')+1-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})\big)},
\end{aligned}
\tag{5.24}
$$

where $R_1^2, R_2^3, R_3, R_4, R_5^1, R_6^1, R_7^1, R_8^2, R_9, \alpha, \beta, \lambda, \mu_1$, and μ_2 are given in Table 5.4.

Proof. The identity in (5.24) corresponds to a case of scenario 8 of Table 5.3 ending with a diagonal preceded by an arrival. We present the summary of the event and the transitions of $M/C_2/1$ system in vacation period (τ), in phase 1 service (t_1), and in phase 2 service (t_2) in Table 5.9. Following the proof of Theorem 5.4.9, we construct appropriate Erlang distribution for vacation period, (t_1) and (t_2) and present them in last row of Table 5.9. Here $N_3 = m+n-2d-k_0-s(k-k')-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})+1$ and $N_4 = \sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})+d-1$. By integrating over the product of each Erlang distribution and the corresponding number of lattice paths, we prove 5.24.

□

Table 5.9: Transitions for $M/C_2/1$ for vacation period, t_1 and t_2: The system ends with diagonal preceded by an arrival

Event	**Vacation period**	t_1	t_2
Number of arrivals	$s(k-k')$	$m-d-k_0-s(k-k')$ $-\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})$	$\sum_{v=1}^{d_h}(l_{i_v}-p_{i_v})$
Number of departure after phase 1	-	$n-2d+1$	-
Number of entries into phase 2	-	d	-
Number of departure after phase 2	-	-	$d-1$
Total number of transition	$s(k-k')$	N_3	N_4
Erlang distribution	$\frac{e^{-\lambda\tau}\lambda^{s(k-1)}\tau^{sk}}{\Gamma(sk+1)}$	$\frac{e^{-(\lambda+\mu_1)t_1}(\lambda+\mu_1)^{N_3}}{\Gamma(N_3)}$ $\times t_1^{N_3-1}$	$\frac{e^{-(\lambda+\mu_2)(t-\tau-t_1)}(\lambda+\mu_2)^{N_4}}{\Gamma(N_4)}$ $\times(t-\tau-t_1)^{N_4-1}$

Combining theorems 5.3.7., 5.3.8., 5.3.9., 5.3.10., 5.3.11., 5.3.12., 5.3.13, and 5.3.14 we have the following main theorem.

Theorem 5.3.15. (Probability density function of IBP of $M/C_2/1$ system under (k', k) control policy)

Let $f_{k_0;k',k}(t)$ denote the probability density function of IBP of $M/C_2/1$ system starting initially with k_0 customers at time t under (k', k) policy. Then we have

$$f_{k_0;k',k}(t) = \sum_{i=1}^{8} f^i_{k_0;k',k}(t). \tag{5.25}$$

5.4 Numerical Performance and Comments

To compute the performance of the density function of incomplete busy period of equation (5.25), we wrote a program in R encompassing the lattice path structure. The corresponding code can be found in Appendix A3. The program starts with generating all possible lattice paths using the library AlgDesign. Next, only the paths satisfying the SPLDE, SPLAEV, SPLAEB and SPLDgE are filtered. These paths form the set L and finally equation (5.25) is computed for the selected paths.

The code for computing the density is presented in Appendix A2. On a computation process for $t = 0.2(0.01)4$, the computation takes < 14 CPU minutes.

The output presented in Figure 5.5-Figure 5.8. From Figure 5.5, $0 < t \leq 4$ we notice that as t increases the density function of incomplete busy period $f_k(t)$ computed for $t = 0.2(0.01)4$ and different λ increases. Then, decreases after attaining a maximum value, i.e. satisfying the expected pattern at a larger unit of time we expect the density function to cease, leading to zero probability at such time i.e. $t = 4$, thus justifying our results.

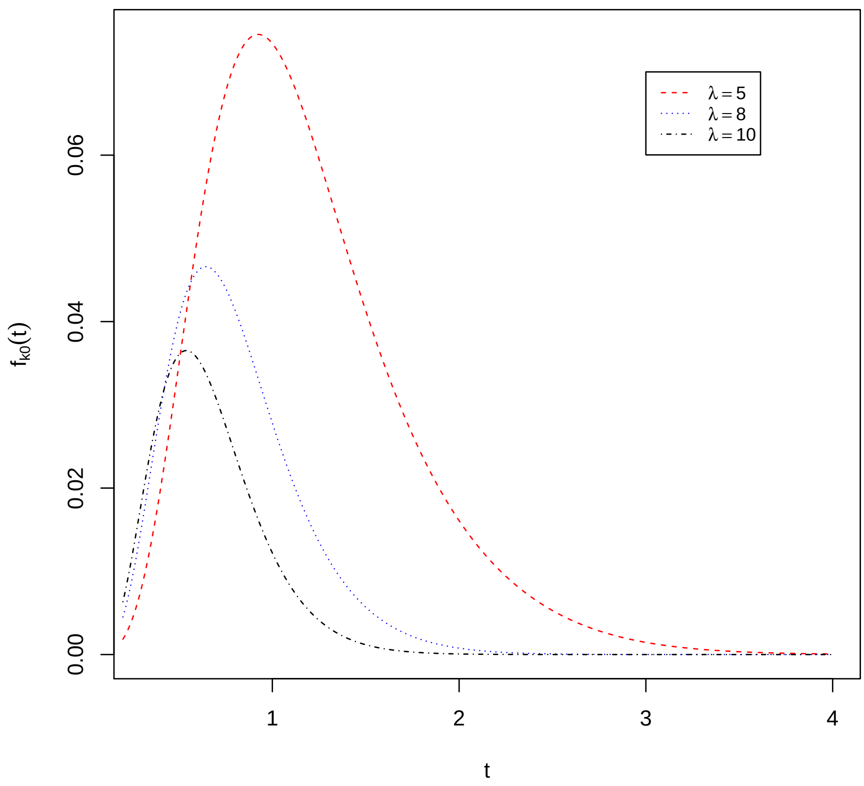


Figure 5.5: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k',k}(t)$, for different λ taking $k_0 = 2,\ k' = 1,\ k = 3,\ \mu_1 = 8,\ \mu_2 = 4,\ \alpha = 0.4,\ \beta = 0.6$

It can be seen from Figure 5.6 that for different values of μ_1, we notice that as t increases, the density function of IBP system increases. Then, after attaining a maximum value, the $f_k^1(t)$ decreases. The rate of increase for smaller values of μ_1 is less than for larger values of μ_1 and the rate of decrease for smaller values of μ_1 is less than for larger values of μ_1.

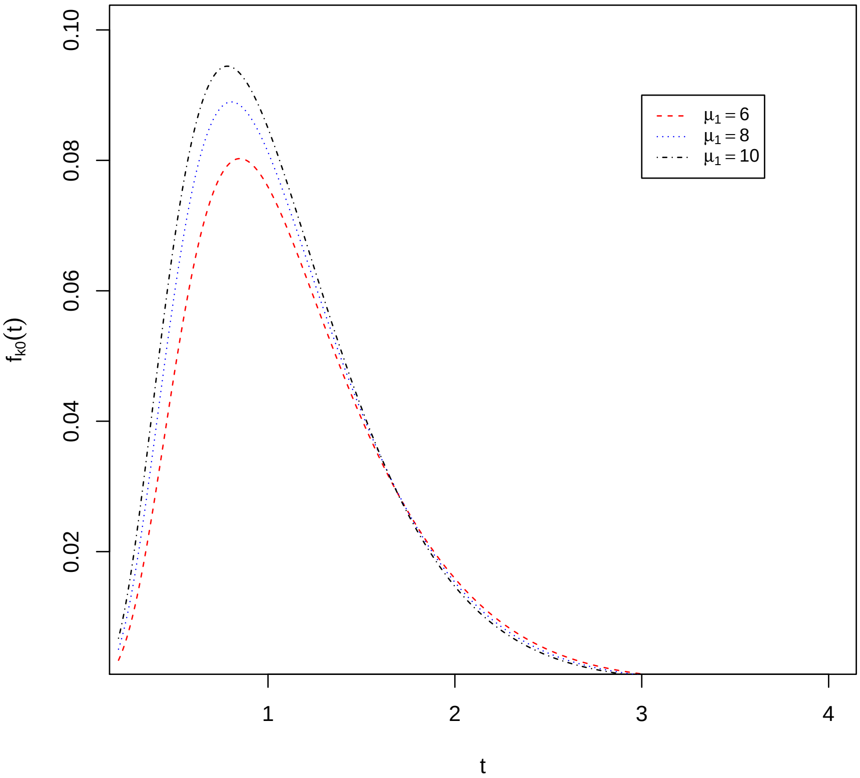


Figure 5.6: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k',k}(t)$, for different μ_1 taking $k_0 = 2,\ k' = 1,\ k = 3,\ \lambda = 5,\ \mu_2 = 4,\ \alpha = 0.4,\ \beta = 0.6$

Figure 5.7 shows that for different values of μ_2, as t increases, the density function of IBP system increases and after attaining a maximum, it decreases. As μ_2 increases, the density also increases. But for a particular value of t, we notice that IBP system increases with a decrease in the value of μ_2.

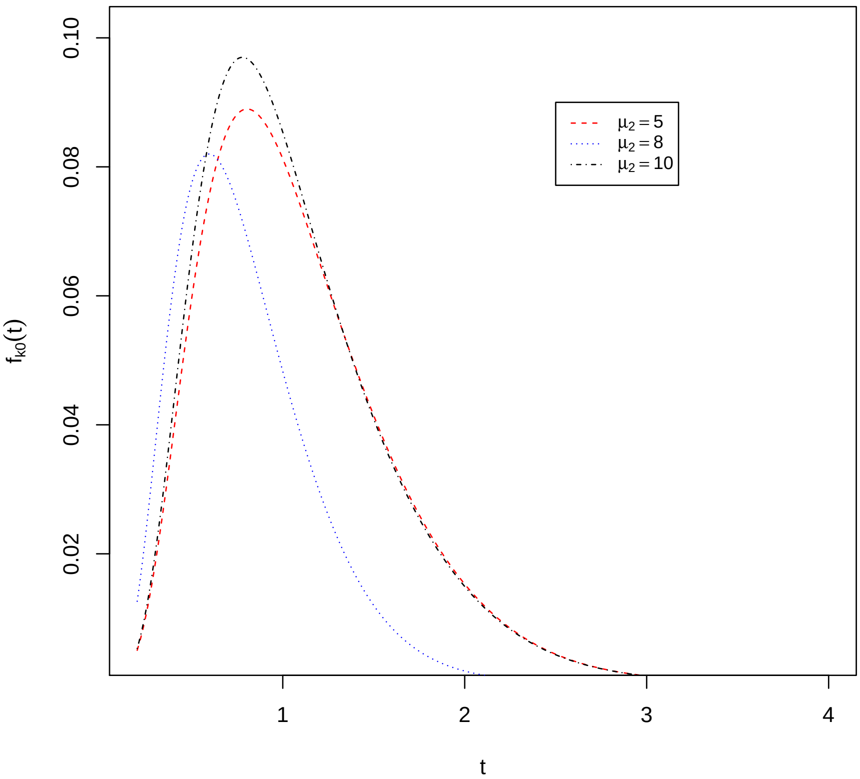


Figure 5.7: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k',k}(t)$, for different μ_2 taking $k_0 = 2,\ k' = 1,\ k = 3,\ \lambda = 5,\ \mu_1 = 8,\ \alpha = 0.4,\ \beta = 0.6$

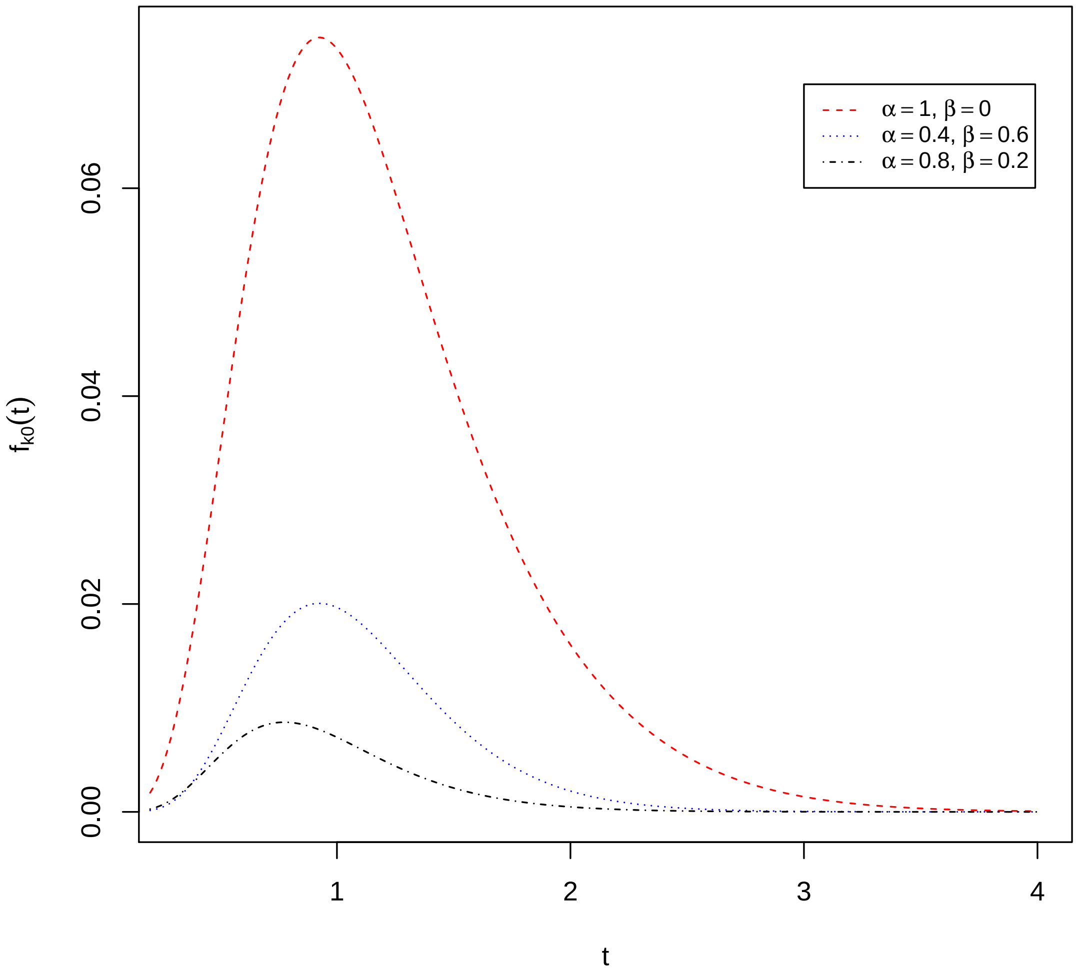


Figure 5.8: Density of the incomplete busy period system of $M/C_2/1$ queue, $f_{k_0;k',k}(t)$, for different α and β taking $k_0 = 1,\ k = 2,\ \lambda = 5,\ \mu_1 = 8,\ \mu_2 = 4$

Figure 5.8 shows that for different values of α and β, as t increases, the density function of IBP system initially increases, then decreases. An increase in α means more and more customers enter the next phase of service. The rate of decrease in the value of density for smaller values of α is greater than that for the larger value of α.

The computation of density function via R is efficient and quick. This can be used as input into the other systems.

5.5 Summary

In this chapter, we develop the incomplete busy period density function of a $M/G/1$ queues under (k', k) control policy through LPC approach which entails approximating general service distribution by Coxian 2-phase distribution. This development includes

(i) Constructing a complete set of structural properties for lattice path representations with last departure, last arrival and last diagonal events. This includes defining the basic notation and terminology, representing the incomplete busy period, constructing the run, inserting the diagonals, and approximating the service time by Coxian 2-phase distribution.

(ii) Deriving the enumeration of the number of lattice paths corresponding to its structural properties.

(iii) Obtaining the incomplete busy period density function of M/G/1 queues under (k', k) control policy

(iv) Developing computer codes to estimate the busy period densities for above models. This includes generating all possible lattice paths using the library AlgDesign, next only the paths satisfying conditions of structural properties are filtered. These paths form the set L and finally the density function is derived for the selected paths. The required algorithms and the computational procedures are developed as R programs (Team [97]).

The number of lattice paths when the system ends with departure, arrival in vacation period, arrival in phase 1 of service, arrival in phase 2 of service, a diagonal preceded a departure and a diagonal preceded an arrival can be counted using (5.3.1), (5.3.2), (5.3.3), (5.3.4), (5.3.5), and (5.3.6), respectively. The number of lattice paths in which no diagonal inserted can be considered as special case i.e $p = 0$ and can be counted using (5.3.7). Furthermore the explicit form of transient analysis of the incomplete busy period is given in (5.25).

The investigation of the influence of taking different values of parameters (λ, μ_1, μ_2, α, β) on the behaviour of the graphs of the density function was presented in

Figures 5.5 - 5.8. The graphs indicate that as t increases, the density function of IBP system $f_{k_0}(t)$ computed for $t = 0.2(0.01)4$ decreases.

Taking k'=0, (5.25) reduces to the probability density function of IBP of $M/C_2/1$ system under $(0, k)$ control policy (4.24).

Chapter 6

CONCLUSIONS AND FUTURE WORK

6.1 Summary

In this thesis we focus on developing the density function of pure incomplete busy period (PIBP) for $M/G/1$ queuing systems and incomplete busy period (IBP) for such queueing systems under control policies. The $M/G/1$ queueing systems refer to those systems in which the arrivals are generated by a Poisson process but the service time is permitted to be non-exponential distribution. Hence, there is no explicit distribution for the service time but we assume that the service times of all customers are independently and identically distributed (i.i.d).

We recapitulate the main focus of this thesis i.e.

1. to develop the PIBP of $M/G/1$ queues,
2. to develop the IBP of $M/G/1$ queues, operating under the following two control policies:
   (i) $(0, k)$ control policy where the server goes on vacation when the system becomes empty and re-opens for service when the kth customer arrives at the system,
   (ii) $(k, k^{'})$ control policy where the server starts serving only when the number of customers in the queue becomes k and remains busy as long as there are at least customers k' waiting for service.

Our research method is based on extending the lattice path approach (Agarwal et al. [1, 2]) to analyse the output characteristics of the system while the service distribution is approximated by (C_2), Coxian 2-phase distribution.

Some of the major steps that are used in developing our results are:

- Representing the queuing process as appropriate lattice paths.
- Define the structural properties of lattice paths for each queueing model scenario.
- Counting the number of lattice paths for each scenario.
- Finding the probability associated with the lattice paths presenting for each scenario to derive the corresponding density function.
- Writing computer codes (programs) in R (Team [97]) to compute the busy period density function for above models.
- Validating the results by performing the numerical computations for different sets of values of the parameters involved and presenting practical guidelines for efficient computation.

In this thesis we have derived the following key results.

1. Developing the pure incomplete busy period (PIBP) density function of $M/G/1$ queues.
   (i) The structural properties for the problem have been formulated and corresponding density is computed based on lattice path approach. Further an R-code is developed for computing the density.
   (ii) Based on the mathematical models constructed and the solutions obtained, a number of sensitivity analysis has been carried out to study the influence of taking different values of parameters on the behaviour of the graphs of the density function. It shows that as t increases, the density function of IBP system $f_{k_0}(t)$ computed for $t = 0.005(0.1)1.5$ decreases.

2. Developing the incomplete busy period (IBP) density function of $M/G/1$ queues under $(0,k)$ control policy, wherein the server goes on the vacation when the system becomes empty and re-opens for service immediately at the arrival of the k^{th} customer.

   (i) The structural properties for the problem have been formulated and corresponding density is computed based on lattice path approach. Further an R-code is developed for computing the density.

   (ii) Based on the mathematical models constructed and the solutions obtained, the sensitivity of taking different values of parameters on the behaviour of the graphs of the density function is investigated. It shows that as t increases, the density function of IBP system $f_{k_0;k}(t)$ computed for $t = 0.03(0.01)4$ decreases.

3. Developing the incomplete busy period (IBP) density function of $M/G/1$ queues under $(k,k^{'})$ control policy, wherein the server starts serving only when the number of customers in the queue becomes k and remains busy as long as there are at least $k^{'}$ customers waiting for service.

   (i) The structural properties for the problem have been formulated and corresponding density is computed based on lattice path approach. Further an R-code is developed for computing the density.

   (ii) The investigation of the influence of taking different values of λ on the behaviour of the graphs of the density function shows that as t increases, the density function of IBP system $f_{k_0;k',k}(t)$ computed for $t = 0.2(0.01)4$ decreases.

6.2 Significance

This research leads to significant advancements of the use of the LPC to compute transient probabilities for $M/G/1$ queueing systems and such systems operated under $(0,k)$ and $(k,k^{'})$ control policies. Counting of lattice paths generated by $M/G/1$ queuing systems and probability density function for its PIBP are presented in theorems 3.4.1 - 3.4.5. Furthermore counting of lattice paths generated by such systems and probability density function for its IBP under $(0, k)$ control policy are presented in theorems 4.3.1 - 4.3.15 while for its IBP under (k, k') are presented in theorems 5.3.1 - 5.3.15.

The general service time is approximated by C_2, Coxian 2-phase distribution that has Markovian property, enabling us to represent the processes by two-dimensional LPs. As distributions C_2 cover a wide class of distribution that have rational Laplace-Stieltjes transform (LST) and square coefficient of variation (CV^2) lying in $[1/2, \infty)$, the results obtained are applicable to a large class of real life situation.

Furthermore the density functions obtained can be computed via R-codes developed. The computation code presented can be used as an input in the optimization process for developing optimal strategies to effectively manage a queueing system.

6.3 Further Research

In this thesis, we use lattice path approach to develop incomplete busy period density function of $M/G/1$ queueing systems. Although some important results have been obtained, there are still open problems that need further research. These are

- to extend LPC results for queueing models with general arrivals, batch service and/or arrivals.
- to develop incomplete busy period density function of $M/G/1$ queueing systems with server breakdowns and multiple vacations.
- to develop incomplete busy period density function under $(0, k)$ policy with startup/closedown.
- Sensitivity analysis for time taken for calculating the probability density function of $M/G/1$ queueing systems parameter.

Appendix A

R Codes for Estimating Probability Density Function

Soft copy of the R-code can be obtained by making a written request to the Department of Mathematics and Statistics, Curtin University, GPO Box U1987 Perth, WA 6845.

A.1 R Codes for Estimating The Pure Incomplete Busy Period (PIBP) Density Function

```
#----------------------------------------------------------------------------
#File name: Chapter 3 density function
#Aim: Graphing the density function
#----------------------------------------------------------------------------


#----------------------------------------------------------------------------
#Function First Component: fk1
#Aim: calculating the first part of the density function
#Input: lambda1,beta1,mu1,mu2,sumfk1, k0
#Output: the calculation of the first part of the density function
#----------------------------------------------------------------------------


t <- as.matrix(seq(0.005,1.5,0.1))
lambda1=5
beta1=0.6
mu1=8
```

```
sumfk1=0
k0=1

for (m in 2:50) {
  for (n in 0:(m-1))
  {
    a11=choose(m+n-k0,n)-choose(m+n-k0,m)
    a12=lambda1^(m-k0)
    a13=(beta1*mu1)^n
    a14=(t^(m+n-k0-1))/factorial(m+n-k0-1)
    a15=exp((-lambda1-mu1)*t)
    a1=a11*a12*a13*a14*a15
    sumfk1=sumfk1+a1
  }
}


#-----------------------------------------------------------------------------
#Function factMatrix
#Aim: getting all possible pairs of l's and L's based on the possible values of
#l's and L's
#Input: m,n,r1
#Output: all possible pairs of l's and L's
#-----------------------------------------------------------------------------

factMatrix1 <- function(m,n,r1) {
    library(AlgDesign)
    #lengths contains the lengths of l1, l2,...L1,L2,... in that order
    #first and last contain the first and last indices of  l1,l2,.,L1,L2,.
    in that order
    lengths <-rep(0,2*r1)
    print(c("lengths="))
    print(lengths)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    #get first and last entry of array lengths
    lengths[1] <- m-3
    first[1] <- 4
    last[1] <- m
    lengths[2*r1] <- n
    first[2*r1] <- 1
    last[2*r1] <- n

    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
    }
    if (r1 > 1 ) {
```

```
        for (i in (r1+1):(2*r1-1)) {
          lengths[i] <- n
          first[i] <- 1
          last[i] <- n

        }
    }

    #generate design matrix
    design <- gen.factorial(lengths)
    #design <- gen.factorial(2,3,1)
    design <- design - max(design +1)
    #print(c("design"))
    #print(design)
    #recode design matrix
    for (i in 1:(2*r1)) {
        #extract the unique values in column i
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
        }
    }

    design
}

factMatrix2 <- function(m,n,r1) {
    library(AlgDesign)
    lengths <-rep(0,2*r1)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    lengths[1] <- m-2
    first[1] <- 3
    last[1] <- m
    lengths[2*r1] <- n
    first[2*r1] <- 1
    last[2*r1] <- n
    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
    }
    if (r1 > 1 ) {
        for (i in (r1+1):(2*r1-1)) {
          lengths[i] <- n
          first[i] <- 1
          last[i] <- n
```

```
        }
    }
    #generate design matrix
    design <- gen.factorial(lengths)
    design <- design - max(design +1)
    for (i in 1:(2*r1)) {
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
        }
    }
    design
}

factMatrix3 <- function(m,n,r1) {
    library(AlgDesign)
    lengths <-rep(0,2*r1)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    lengths[1] <- m-4
    first[1] <- 5
    last[1] <- m
    lengths[2*r1] <- n
    first[2*r1] <- 1
    last[2*r1] <- n
    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
    }
    if (r1 > 1 ) {
        for (i in (r1+1):(2*r1-1)) {
          lengths[i] <- n
          first[i] <- 1
          last[i] <- n
        }
    }

    #generate design matrix
    design <- gen.factorial(lengths)
    design <- design - max(design +1)
    for (i in 1:(2*r1)) {
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
```

```
        }
    }
    design
}

factMatrix4 <- function(m,n,r1) {
    library(AlgDesign)
    lengths <-rep(0,2*r1)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    lengths[1] <- m-3
    first[1] <- 4
    last[1] <- m
    lengths[2*r1] <- n+1
    first[2*r1] <- 0
    last[2*r1] <- n
    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
    }
    if (r1 > 1 ) {
        for (i in (r1+1):(2*r1-1)) {
          lengths[i] <- n
          first[i] <- 1
          last[i] <- n
        }
    }
    #generate design matrix
    design <- gen.factorial(lengths)
    design <- design - max(design +1)
    for (i in 1:(2*r1)) {
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
        }
    }
    design
}

factMatrix5 <- function(m,n,r1) {
    library(AlgDesign)
    lengths <-rep(0,2*r1)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    lengths[1] <- m-2
    first[1] <- 3
```

```
    last[1] <- m
    lengths[2*r1] <- n+1
    first[2*r1] <- 0
    last[2*r1] <- n
    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
    }
    if (r1 > 1 ) {
        for (i in (r1+1):(2*r1-1)) {
          lengths[i] <- n
          first[i] <- 1
          last[i] <- n
        }
    }
    #generate design matrix
    design <- gen.factorial(lengths)
    design <- design - max(design +1)
    for (i in 1:(2*r1)) {
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
        }
    }
    design
}


#------------------------------------------------------------------------------
#Function2
#Aim: getting pairs of l's and L's  satisfying the condition
#Input: design,m,n,p,r1
#Output:all possible pairs of l's and L's satisfying the condition
#------------------------------------------------------------------------------

function2<-function(design,m,n,p,r1){
    count.no.rows=0
    design.subset<-design
    for(i in 1:dim(design)[1]) {
        cond=TRUE
        sum1=0
        sum2=0
        for (j in 1:r1) {
          sum1<-sum1+design[i,j]
          sum2<-sum2+design[i,j+r1]
          cond<-(cond&&(sum1>sum2))
```

```
        }
        cond<-(cond&&(sum1==m-p)&&(sum2==n-p))
        if (cond) {
            count.no.rows=count.no.rows+1
            design.subset[count.no.rows,]<-design[i,]
        }
    }
    design.subset[1:count.no.rows,]
}


#-------------------------------------------------------------------------------
#Function3
#Aim: calculating the summation
#Input: m,n,k0,p,q1,r1,t,sumsq
#Output: the calculation of last summation part of second term, third term, etc
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#fk0;2(t)
#-------------------------------------------------------------------------------
function32<-function(m,n,k0,p,q1,r1,t,sumsq) {
    f11=choose(n-p-r1,p-q1)
    f12=(alpha1^p)*(beta1^(n-2*p))*(lambda1^(m-p-k0))*(mu1^(n-p))*(mu2^p)*
        exp((-lambda1-mu1)*t)
    f1=f11*f12
    sum11=0
    for (x in 0:50) {
        if (m+n-(2*p)+x-1-sumsq < 1) {
            sum11<-0
            break
        }
        f21=(((mu2-mu1)^x)/factorial(x))
        f22= t^(m+n-p-k0+x-1)/factorial(m+n-p-k0+x-1)
        f23=factorial(m+n-(2*p)-k0+x-1-sumsq)/factorial(m+n-(2*p)-k0-1-sumsq)
        sum11=sum11+(f21*f22*f23)
    }
    f2=f1*sum11
    f2
}


#-------------------------------------------------------------------------------
#fk0;3(t)
#-------------------------------------------------------------------------------
function33<-function(m,n,k0,p,q1,r1,t,sumsq) {
    f11=choose(n-p-r1+1,p-q1)
    f12=(alpha1^p)*(beta1^(n-2*p))*(lambda1^(m-p-k0))*(mu1^(n-p))*(mu2^p)*
    exp((-lambda1-mu1)*t)
    f1=f11*f12
    sum11=0
```

```
    for (x in 0:50) {
        if (m+n-(2*p)+x-k0-sumsq < 1) {
            sum11<-0
            break
        }
        f21=(((mu2-mu1)^x)/factorial(x))
        f22= t^(m+n-p-k0+x-1)/factorial(m+n-p-k0+x-1)
        f23=factorial(m+n-(2*p)-k0+x-sumsq)/factorial(m+n-(2*p)-k0-sumsq)
        sum11=sum11+(f21*f22*f23)
    }
    f2=f1*sum11
    f2
}


#-------------------------------------------------------------------------------
#fk0;4(t)
#-------------------------------------------------------------------------------
function34<-function(m,n,k0,p,q1,r1,t,sumsq) {
    f11=choose(n-p-r1+1,p-q1)
    f12=(alpha1^p)*(beta1^(n-2*p+1))*(lambda1^(m-p-k0))*(mu1^(n-p+1))
        *(mu2^(p-1))*exp((-lambda1-mu1)*t)
    f1=f11*f12
    sum11=0
    for (x in 0:50) {
        if (m+n-(2*p)+x-k0-sumsq < 1) {
            sum11<-0
            break
        }
        f21=(((mu2-mu1)^x)/factorial(x))
        f22= t^(m+n-p-k0+x-1)/factorial(m+n-p-k0+x-1)
        f23=factorial(m+n-(2*p)-k0+x-sumsq)/factorial(m+n-(2*p)-k0-sumsq)
        sum11=sum11+(f21*f22*f23)
    }
    f2=f1*sum11
    f2
}


#-------------------------------------------------------------------------------
#fk0;5(t)
#-------------------------------------------------------------------------------
function35<-function(m,n,k0,p,q1,r1,t,sumsq) {
    f11=choose(n-p-r1,p-q1-1)
    f12=(alpha1^p)*(beta1^(n-2*p+1))*(lambda1^(m-p-k0))*(mu1^(n-p+1))
        *(mu2^(p-1))*exp((-lambda1-mu1)*t)
    f1=f11*f12
    sum11=0
    for (x in 0:50) {
        if (m+n-(2*p)+x-k0-sumsq < 1) {
            sum11<-0
```

```
            break
        }
        f21=(((mu2-mu1)^x)/factorial(x))
        f22= t^(m+n-p-k0+x-1)/factorial(m+n-p-k0+x-1)
        f23=factorial(m+n-(2*p)-k0+x-sumsq)/factorial(m+n-(2*p)-k0-sumsq)
        sum11=sum11+(f21*f22*f23)
    }
    f2=f1*sum11
    f2
}


#-------------------------------------------------------------------------------
#fk0;6(t)
#-------------------------------------------------------------------------------
function36<-function(m,n,k0,p,q1,r1,t,sumsq) {
    f11=choose(n-p-r1+1,p-q1)
    f12=(alpha1^p)*(beta1^(n-2*p))*(lambda1^(m-p-k0))*(mu1^(n-p+1))
        *(mu2^(p-1))*exp((-lambda1-mu1)*t)
    f1=f11*f12
    sum11=0
    for (x in 0:50) {
        if (m+n-(2*p)+x-k0-sumsq < 1) {
            sum11<-0
            break
        }
        f21=(((mu2-mu1)^x)/factorial(x))
        f22= t^(m+n-p-k0+x-1)/factorial(m+n-p-k0+x-1)
        f23=factorial(m+n-(2*p)-k0+x-sumsq)/factorial(m+n-(2*p)-k0-sumsq)
        sum11=sum11+(f21*f22*f23)
    }
    f2=f1*sum11
    f2
}

#-------------------------------------------------------------------------------
#Function4
#Aim: getting design of R8
#Input: lis
#Output: the calculation of (l(is)-p(is))
#-------------------------------------------------------------------------------

        designR8<-function(lis){
        len.lis <- length(lis)
        #print(c("len.lis=", len.lis))
        #get arguments for gen.factorial
        gen.fact <- paste("gen.factorial(c(1,",lis[1])
          if (len.lis >= 2) {
            for (i in 2:len.lis) {
            gen.fact <- paste(gen.fact,",1,",lis[i]+1)
            }
```

```
          }
        gen.fact <- paste(gen.fact,"))")
        print(c("gen.fact=",gen.fact))
        #columns of fact are l1s, p1s, l2s, p2s, l3s, p3s, ...
        fact <- eval(parse(text=gen.fact))
        print(c("fact=",fact))
        fact <- fact - max(fact + 1)
        print(c("fact=",fact))
        #coding given in terms -1 0 1 etc so need to recode
        fact.ncols <- dim(fact)[2]
        j<-1
        for (i in 1:len.lis) {
          #relabel
          fact[,j] <- lis[i]
          j <- j+1
          #extract unique values in column i
          u <- sort(unique(fact[,j]),decreasing=TRUE)
          if (i == 1) {
            for (l in seq(1,lis[i])) {
                fact[fact[,j] == u[l],j] <- l
            }
          }
          else {
             for (l in seq(0,lis[i])) {
                fact[fact[,j] == u[l+1],j] <- l
            }
          }
            j <- j+1
        }
    fact
  }



#-------------------------------------------------------------------------------
#Function5 for fk2
#Aim: calculating the second part of the density function
#Input:lambda1, alpha1, beta1, mu1, mu2, k0  and all other functions
#Output: the calculation of second part
#-------------------------------------------------------------------------------

t <- as.matrix(seq(0.005,1.5,0.1))
lambda1=5
alpha1=0.4
beta1=0.6
mu1=8
mu2=4
k0=1

for (m in 4:6){
  for (n in 3:(m-1)){
```

```
    for (p in 1:min(floor(n/2),(m-1))){
     for (r1 in 1:max(min(m-p-1,n-p-1),1)){
        sumfk2=0
        for (q1 in max(0,2*p-m+r1):(min(r1,p))) {
            design.subset<-function2(factMatrix1(m,n,r1),m,n,p,r1)
            if (q1 != 0) {
               num.rows<-dim(design.subset)[1]
               cmb<-combn(c(1:r1),q1)
               for (r in 1:num.rows){
                 for (g in 1:dim(cmb)[2]){
                    indices<-cmb[,g]
                    lis<-design.subset[r,indices]
                    is.numeric(lis)
                    if (length(lis)> 1) lis<-as.numeric(lis)
                    fact <- designR8(lis)
                    fact.nrows <- dim(fact)[1]
                    for (i in 1:fact.nrows){
                    sumsq <- 0
                      for (j in seq(1,dim(fact)[2],2)) {
                      sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                      }
                      sumsq
                      sumfk2=sumfk2+function32(m,n,k0,p,q1,r1,t,sumsq)
                    }
                 }
               }
            }
            sumfk2
            if (q1==0)
                sumfk2=sumfk2+function32(m,n,k0,p,q1,r1,t,0)
        }
        sumfk2
      }
    }
  }
}

for (m in 3:6){
  for (n in 2:(m-1)){
    for (p in 1:min(floor(n/2),(m-1))){
     for (r1 in 1:max(min(m-p-1,n-p-1),1)){
        sumfk3=0
        for (q1 in max(0,2*p-m+r1):(min(r1,p))) {
            design.subset<-function2(factMatrix2(m,n,r1),m,n,p,r1)
            if (q1 != 0) {
               num.rows<-dim(design.subset)[1]
               cmb<-combn(c(1:r1),q1)
               for (r in 1:num.rows){
                 for (g in 1:dim(cmb)[2]){
                    indices<-cmb[,g]
                    lis<-design.subset[r,indices]
```

```
                    is.numeric(lis)
                    if (length(lis)> 1) lis<-as.numeric(lis)
                    fact <- designR8(lis)
                    fact.nrows <- dim(fact)[1]
                    for (i in 1:fact.nrows){
                    sumsq <- 0
                      for (j in seq(1,dim(fact)[2],2)) {
                      sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                      }
                      sumsq
                      sumfk3=sumfk3+function33(m,n,k0,p,q1,r1,t,sumsq)
                    }
                 }
               }
            }
            sumfk3
            if (q1==0)
                sumfk3=sumfk3+function33(m,n,k0,p,q1,r1,t,sumsq)
        }
        sumfk3
      }
    }
  }
}

for (m in 4:6){
  for (n in 3:(m-1)){
    for (p in 1:min(floor(n/2),(m-1))){
     for (r1 in 1:max(min(m-p-1,n-p-1),1)){
        sumfk4=0
        for (q1 in max(0,2*p-m+r1):(min(r1,p))) {
            design.subset<-function2(factMatrix3(m,n,r1),m,n,p,r1)
            if (q1 != 0) {
               num.rows<-dim(design.subset)[1]
               cmb<-combn(c(1:r1),q1)
               for (r in 1:num.rows){
                 for (g in 1:dim(cmb)[2]){
                    indices<-cmb[,g]
                    lis<-design.subset[r,indices]
                    is.numeric(lis)
                    if (length(lis)> 1) lis<-as.numeric(lis)
                    fact <- designR8(lis)
                    #get the sum
                    fact.nrows <- dim(fact)[1]
                    for (i in 1:fact.nrows){
                    sumsq <- 0
                      for (j in seq(1,dim(fact)[2],2)) {
                      sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                      }
                      sumsq
                      sumfk4=sumfk4+function34(m,n,k0,p,q1,r1,t,sumsq)
```

```
                  }
               }
             }
          }
          sumfk4
          if (q1==0)
              sumfk4=sumfk4+function34(m,n,k0,p,q1,r1,t,sumsq)
      }
      sumfk4
    }
  }
 }
}

for (m in 5:6){
  for (n in 4:(m-1)){
    for (p in 1:min(floor(n/2),(m-1))){
     for (r1 in 1:max(min(m-p-1,n-p-1),1)){
        sumfk5=0
        for (q1 in max(0,2*p-m+r1):(min(r1,p))) {
            design.subset<-function2(factMatrix1(m,n,r1),m,n,p,r1)
            if (q1 != 0) {
               num.rows<-dim(design.subset)[1]
               cmb<-combn(c(1:r1),q1)
               for (r in 1:num.rows){
                 for (g in 1:dim(cmb)[2]){
                    indices<-cmb[,g]
                    lis<-design.subset[r,indices]
                    is.numeric(lis)
                    if (length(lis)> 1) lis<-as.numeric(lis)
                    fact <- designR8(lis)
                    #get the sum
                    fact.nrows <- dim(fact)[1]
                    for (i in 1:fact.nrows){
                    sumsq <- 0
                      for (j in seq(1,dim(fact)[2],2)) {
                      sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                      }
                      sumsq
                      sumfk5=sumfk5+function35(m,n,k0,p,q1,r1,t,sumsq)
                    }
                 }
               }
            }
            sumfk5
            if (q1==0)
                sumfk5=sumfk5+function35(m,n,k0,p,q1,r1,t,sumsq)
        }
        sumfk5
      }
    }
```

```
  }
}

for (m in 4:6){
  for (n in 2:(m-1)){
    for (p in 1:min(floor(n/2),(m-1))){
     for (r1 in 1:max(min(m-p-1,n-p-1),1)){
        sumfk6=0
        for (q1 in max(0,2*p-m+r1):(min(r1,p))) {
            design.subset<-function2(factMatrix4(m,n,r1),m,n,p,r1)
            if (q1 != 0) {
               num.rows<-dim(design.subset)[1]
               cmb<-combn(c(1:r1),q1)
               for (r in 1:num.rows){
                 for (g in 1:dim(cmb)[2]){
                    indices<-cmb[,g]
                    lis<-design.subset[r,indices]
                    is.numeric(lis)
                    if (length(lis)> 1) lis<-as.numeric(lis)
                    fact <- designR8(lis)
                    #get the sum
                    fact.nrows <- dim(fact)[1]
                    for (i in 1:fact.nrows){
                    sumsq <- 0
                      for (j in seq(1,dim(fact)[2],2)) {
                      sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                      }
                      sumsq
                      sumfk6=sumfk6+function36(m,n,k0,p,q1,r1,t,sumsq)
                    }
                 }
               }
            }
            sumfk6
            if (q1==0)
                sumfk6=sumfk6+function36(m,n,k0,p,q1,r1,t,sumsq)
        }
        sumfk6
      }
    }
  }
}


fk_t=sumfk1+sumfk2+sumfk3+sumfk4+sumfk5+sumfk6
print(c("fk_t="))
print(fk_t)
plot(t,fk_t,type="l",lty=2,lwd=1,ylab="Density",col="red",xlim=c(0,1.5),
ylim=c(0,6), font.main=1,col.main="red")
```

A.2 R Codes for Estimating IBP Density Function Under $(0,k)$ Control Policy

```
#-----------------------------------------------------------------------------#
            #IBP density function under (0,k) control policy#
                    #Aim: Graphing its density function#
#-----------------------------------------------------------------------------#

#------------------------------------------------------------------------------
#Name of function: f1k0
#Aim: to calculate f1k0(t)
#Input: lambda1, beta1, mu1, sumf1k, k0 and k
#Note: We denote kstar with ks
#Output: the calculation of f1k0(t)
#------------------------------------------------------------------------------

lambda1=5
beta1=0.6
mu1=4
sumf1k=0
k0=2
k=3


t <- as.matrix(seq(0.2,4,0.01))
for (m in 5:10){
  for (ks in 0:(k-1)) {
    for (n in 2:(m-ks)){
      for (s1 in 1: floor(((n-k0)/k)+1)){
        if (((2*n)-k0-((s1-1)*k)) < 1) {
          sumf1k<-0
          break
        }

        if (n-k0-((s1-1)*k)< 0) {
          sumf1k<-0
          break
        }

        a11=exp((-lambda1-mu1)*t)
        a12=(k0+(s1-1)*k)
        a13=(2*n-k0-(s1-1)*k)
        a14=choose((2*n-k0-((s1-1)*k)),n)
        a15=lambda1^(m-k0)
        a16=(beta1^n)/factorial(ks+(s1-1)*k)
        a1=a11*(a12/a13)*a14*a15*a16

          sum21=0
          for (i in 0:30) {
            if ((m+n-k0+i) < 0) {
              sum21<-0
```

```
            break
          }

          f31=((mu1)^(n+i))/factorial(i)
          f32=t^(m+n-k0+i)
          f33=factorial((ks+(s1-1)*k)+i)/factorial(m+n-k0+i)
          sum21=sum21+(f31*f32*f33)
        }
      sum21
      sumf1k=sumf1k+(a1*sum21)
      }
    }
  }
}
f1=sumf1k


#-------------------------------------------------------------------------------
#Name of function: f2k0
#Aim: to calculate f2k0(t)
#Input:lambda1, beta1, mu1, sumf2k, k0, and k
#Output: the calculation of f2k0(t)
#-------------------------------------------------------------------------------

lambda1=5
beta1=0.6
mu1=4
sumf2k=0
k0=2
k=3

t <- as.matrix(seq(0.2,4,0.01))
for (m in 5:10){
  for (n in 2:(m-1)){
    for (s1 in 1: floor((n-k0)/k)){
      if ((m-k0-(s1*k)) < 0) {
        sum11<-0
        break
      }

      if ((n-k0-(s1*k)) < 0) {
        sum11<-0
        break
      }

      a11=exp((-lambda1-mu1)*t)
      a12=choose((m+n-k0-(s1*k)),n)-choose((m+n-k0-(s1*k)),m)
      a13=lambda1^(m-k0)
      a14=(beta1^n)/factorial(s1*k)
      a1=a11*a12*a13*a14
```

```
        sum21=0
        for (i in 0:50) {
          if ((m+n-k0+i+1) < 1) {
              sum21<-0
              break
          }

          f31=((mu1)^(n+i))/factorial(i)
          f32=t^(m+n-k0+i)
          f33=factorial((s1*k)+i)/factorial(m+n-k0+i)
          sum21=sum21+(f31*f32*f33)
        }
    sum21
    sumf2k=sumf2k+(a1*sum21)
    }
  }
}
f2=sumf2k



#------------------------------------------------------------------------------
#Name of function: factMatrix
#Aim: to get all possible pairs of l's and L's based on the possible values of
#l's and L's
#Input: m,n,r1
#Output: all possible pairs of l's and L's based on the possible values of
#l's and L's
#------------------------------------------------------------------------------

factMatrix<- function(m,n,r1) {
    library(AlgDesign)
    lengths <-rep(0,2*r1)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    lengths[1] <- m-1
    first[1] <- 2
    last[1] <- m
    lengths[2*r1] <- n
    first[2*r1] <- 1
    last[2*r1] <- n

    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
    }
    if (r1 > 1 ) {
        for (i in (r1+1):(2*r1-1)) {
```

```
            lengths[i] <- n
            first[i] <- 1
            last[i] <- n

        }
    }

    #generate design matrix
    design <- gen.factorial(lengths)
    design <- design - max(design +1)

    #recode design matrix
    for (i in 1:(2*r1)) {
        #extract the unique values in column i
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
        }
    }
    design
}


#--------------------------------------------------------------------------------
#Name of function: function2
#Aim: getting pairs of l's and L's  satisfying the condition
#Input: design,m,n,d,r1,s1
#Output:all possible pairs of l's and L's satisfying the condition
#--------------------------------------------------------------------------------

function2<-function(design,m,n,d,r1,s1){
    count.no.rows=0
    DSLatticePath<-design
    for(i in 1:dim(design)[1]) {
        cond=TRUE
        sum1=0
        sum2=0
        for (j in 1:r1) {
          sum1<-sum1+design[i,j]
          sum2<-sum2+design[i,j+r1]
          cond<-(cond&&(sum1>=sum2))
        }
        cond<-(cond&&(sum1==m-d)&&(sum2==n-d))
        if (cond) {
            count.no.rows=count.no.rows+1
            DSLatticePath[count.no.rows,]<-design[i,]
        }
    }
    DSLatticePath[1:count.no.rows,]
```

```
}


#---------------------------------------------------------------------------------
#Name of function: function3
#Aim: finding touch and sequence of touchs
#Input: DSLatticePath,r1
#Output: touch and sequence of touchs
#---------------------------------------------------------------------------------

function3<-function(DSLatticePath,r1){
    numberofpaths= dim(DSLatticePath)[1]
    count.touch=matrix(c(rep(0,r1*numberofpaths)), nrow=numberofpaths, ncol=r1)
    for(i in 1:numberofpaths) {
        sum11=0
        sum22=0
        for (j in 1:r1) {
            sum11<-sum11+DSLatticePath[i,j]
            sum22<-sum22+DSLatticePath[i,j+r1]
            if (sum11==sum22) {
                touch<-1
            }
            else {
                touch<-0
            }
             count.touch[i,j]<-touch
        }
    }
  cbind(DSLatticePath,count.touch)
}


#--------------------------------------------------------------------------------
#Name of function: function4
#Aim: to find the position of touching (u_vector)
#Input: DSLatticePath,r1, only rows with a touch or a fealisable lattice path
#       with touch sequence
#Output: u_vector : vector of the position of touching
#--------------------------------------------------------------------------------

function4<-function(arowoffunction3, r1){
    low<-2*r1+1
    high<-3*r1

    ## row getting the touch sequence
    temp<-arowoffunction3
    touchsequence<-(temp[low:high])
    ##counting touches
    no.of.touch<-sum(temp[low:high])
    touchingpositionvector<-rep(0,no.of.touch+1)
```

```
    ##generating ui sequences
    touchingpositionvector[1]<-1

    if (no.of.touch > 0 ) {
        touchingpositionvector[2:(no.of.touch+1)]<-which(touchsequence==1) + 1
        touchingpositionvector
    }
}


#-------------------------------------------------------------------------------
#Name of function: various function5 i.e. function f53k0, f54k0, ..., f57k0
#Aim: to create the R program for various function5.
#The first function5 is f3k0(t)
#Input: m,n,d,dh,r1,t,s1,sumsq
#Output: the function of the summation of f3k0(t)
#-------------------------------------------------------------------------------

function53k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1,d-dh)
    f13=(alpha1^d)*(beta1^(n-2*d))*(lambda1^(m-d-k0))*(mu1^(n-d))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*k)-sumsq < 1) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*k)+i)/factorial(s1*k)
            f35=factorial(m+n-(2*d)-k0-(s1*k)+j-sumsq-1)
            f36=factorial(m+n-(2*d)-k0-(s1*k)-sumsq-1)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f53k0=f1*sum2
}


#-------------------------------------------------------------------------------
#Name of function: function f54k0
#-------------------------------------------------------------------------------

function54k0<-function(m,n,d,dh,r1,t,s1,ks,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
```

```
    f13=(alpha1^d)*(beta1^(n-2*d))*(lambda1^(m-d-k0))*(mu1^(n-d))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-((s1-1)*k)-ks-sumsq-1 < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial(((s1-1)*k)+ks+i)/factorial((s1-1)*k+ks-1)
            f35=factorial(m+n-(2*d)-k0-((s1-1)*k)-ks+j-sumsq-1)
            f36=factorial(m+n-(2*d)-k0-((s1-1)*k)-ks-sumsq-1)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f54k0=f1*sum2
}


#-------------------------------------------------------------------------------
#Name of function: function f55k0
#-------------------------------------------------------------------------------

function55k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-2*d))*(lambda1^(m-d-k0))*(mu1^(n-d))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*k)-sumsq-1 < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i-1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*k)+i)/factorial((s1*k)-1)
            f35=factorial(m+n-(2*d)-k0-(s1*k)+j-sumsq-1)
            f36=factorial(m+n-(2*d)-k0-(s1*k)-sumsq-1)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f55k0=f1*sum2
}
```

```
#------------------------------------------------------------------------------
#Name of function: function f56k0
#------------------------------------------------------------------------------

function56k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-(2*d)+1))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*k)-sumsq < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i-1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*k)+i)/factorial((s1*k)-1)
            f35=factorial(m+n-(2*d)-k0-(s1*k)+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-(s1*k)-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f56k0=f1*sum2
}

#------------------------------------------------------------------------------
#Name of function: function f57k0
#------------------------------------------------------------------------------

function57k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1,d-dh-1)
    f13=(alpha1^d)*(beta1^(n-(2*d)+1))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*k)-sumsq < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i+1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*k)+i)/factorial((s1*k)-1)
```

```
            f35=factorial(m+n-(2*d)-k0-(s1*k)+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-(s1*k)-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f57k0=f1*sum2
}


#-------------------------------------------------------------------------------
#Name of function: function f58k0
#-------------------------------------------------------------------------------

function58k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-(2*d)+1))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*k)-sumsq < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i+1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*k)+i)/factorial((s1*k)-1)
            f35=factorial(m+n-(2*d)-k0-(s1*k)+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-(s1*k)-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f57k0=f1*sum2
}



#-------------------------------------------------------------------------------
#Name of function: function6
#Aim: to get the design of R8
#Input: lis
#Output: the calculation of (l(is)-p(is))
#-------------------------------------------------------------------------------

function6<-function(lis){
    len.lis <- length(lis)
    gen.fact <- paste("gen.factorial(c(1,",lis[1]+1)
```

```
    if (len.lis >= 2) {
        for (i in 2:len.lis) {
            gen.fact <- paste(gen.fact,",1,",lis[i]+1)
        }
    }

    gen.fact <- paste(gen.fact,"))")
    #columns of fact are l1s, p1s, l2s, p2s, l3s, p3s, ...
    fact <- eval(parse(text=gen.fact))
    fact <- fact - max(fact + 1)

    #coding given in terms -1 0 1 etc so need to recode
    fact.ncols <- dim(fact)[2]
    j<-1
    for (i in 1:len.lis) {
        #relabel
        fact[,j] <- lis[i]
        j <- j+1
        #extract unique values in column i
        u <- sort(unique(fact[,j]),decreasing=TRUE)
        if (i == 1) {
            for (l in seq(1,lis[i])) {
                fact[fact[,j] == u[l],j] <- l
            }
        }
        else {
             for (l in seq(0,lis[i])) {
                fact[fact[,j] == u[l+1],j] <- l
             }
        }
        fact[lis[i]==0,j] <- 0
        j <- j+1
    }
    fact
  }



#------------------------------------------------------------------------------
#Input:lambda1,alpha1,beta1,mu1,and mu2
#Output: the calculation of f3k0(t), f4k0(t), f5k0(t), f6k0(t), f7k0(t)
#------------------------------------------------------------------------------

t <- as.matrix(seq(0.2,4,0.01))
lambda1=5
alpha1=0.4
beta1=0.6
mu1=8
mu2=4

#------------------------------------------------------------------------------
```

```
#Name of function: f3
#--------------------------------------------------------------------------------

for (m in 4:6){
  for (n in 2:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-2,n-d-2),2)){
        f3=0
        for (dh in max(0,2*d-m+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]

                for (i in 1:fact.nrows){
                  sumsq <- 0
                  for (j in seq(1,dim(fact)[2],2)) {
                    sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                  }
                  sumsq
                  f3=f3+function53k0(m,n,d,dh,r1,t,s1, sumsq)
                }
              }
```

```
          }
        }
        #sumx
        if (dh==0)
          f3=f3+function53k0(m,n,d,dh,r1,t,s1,0)
      }
      f3
    }
  }
}
}


#---------------------------------------------------------------------------------
#Name of function: function f4
#---------------------------------------------------------------------------------

for (m in 5:6){
  for (n in 2:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-2,n-d-2),2)){
        f4=0
        for (dh in max(0,2*d-m+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)
```

```
              #print(c("len.lis=", len.lis))
              fact <- function6(lis)
              #get the sum
              fact.nrows <- dim(fact)[1]

              for (i in 1:fact.nrows){
                sumsq <- 0
                for (j in seq(1,dim(fact)[2],2)) {
                  sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                }
                sumsq
                f4=f4+function54k0(m,n,d,dh,r1,t,s1, sumsq)
              }
            }
          }
        }
        if (dh==0)
          f4=f4+function54k0(m,n,d,dh,r1,t,s1,0)
      }
     f4
    }
  }
 }
}


#-------------------------------------------------------------------------------
#Name of function: function f5
#-------------------------------------------------------------------------------

for (m in 5:6){
  for (n in 2:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-2,n-d-2),2)){
        f5=0
        for (dh in max(0,2*d-m+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
```

```
          }
        }
        count.no.touch<-function3(DSLatticePath,r1)
        #Calculation of the summation
        if (dh != 0) {
          num.rows<-dim(DSLatticePath)[1]
          cmb<-combn(c(1:r1),dh)
          for (r in 1:num.rows){
            for (g in 1:dim(cmb)[2]){
              indices<-cmb[,g]
              lis<-DSLatticePath[r,indices]
              is.numeric(lis)
              if (length(lis)> 1) lis<-as.numeric(lis)

              #print(c("len.lis=", len.lis))
              fact <- function6(lis)
              #get the sum
              fact.nrows <- dim(fact)[1]

              for (i in 1:fact.nrows){
                sumsq <- 0
                for (j in seq(1,dim(fact)[2],2)) {
                  sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                }
                sumsq
                f5=f5+function55k0(m,n,d,dh,r1,t,s1, sumsq)
              }
            }
          }
        }
        #sumx
        if (dh==0)
          f5=f5+function55k0(m,n,d,dh,r1,t,s1,0)
      }
      f5
    }
  }
 }
}


#--------------------------------------------------------------------------------
#Name of function: function f6
#--------------------------------------------------------------------------------

for (m in 6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-2,n-d-2),2)){
        #we consider k=1 otherwise max(min(m-p-k+1,n-p),1)
        f6=0
```

```
for (dh in max(0,2*d-m+r1):(min(r1,d))) {
  for (s1 in 1:floor((n-k0)/k)) {
    DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
    ####    assume k=1
    DSLatticePath2<-function3(DSLatticePath,r1)
    num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
    for (rowDS in 1:num.Rows.DSLatticePath2) {
      ui.vector <- function4(DSLatticePath2[rowDS,],r1)
      if (length(ui.vector)== s1+1) {
        for (j in 2: (length(ui.vector))) {
          if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
            DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
            ui.vector[j]] - 2
          }
        }
      }
    }
  }
  count.no.touch<-function3(DSLatticePath,r1)
  #Calculation of the summation
  if (dh != 0) {
    num.rows<-dim(DSLatticePath)[1]
    cmb<-combn(c(1:r1),dh)
    for (r in 1:num.rows){
      for (g in 1:dim(cmb)[2]){
        indices<-cmb[,g]
        lis<-DSLatticePath[r,indices]
        is.numeric(lis)
        if (length(lis)> 1) lis<-as.numeric(lis)

        #print(c("len.lis=", len.lis))
        fact <- function6(lis)
        #get the sum
        fact.nrows <- dim(fact)[1]

        for (i in 1:fact.nrows){
          sumsq <- 0
          for (j in seq(1,dim(fact)[2],2)) {
            sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
          }
          sumsq
          f6=f6+function56k0(m,n,d,dh,r1,t,s1, sumsq)
        }
      }
    }
  }
  #sumx
  if (dh==0)
    f6=f6+function56k0(m,n,d,dh,r1,t,s1,0)
}
f6
```

```
      }
    }
  }
}



#--------------------------------------------------------------------------------
#Name of function: function f7
#--------------------------------------------------------------------------------

for (m in 6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-2,n-d-2),2)){
        #we consider k=1 otherwise max(min(m-p-k+1,n-p),1)
        f7=0
        for (dh in max(0,2*d-m+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]
```

```
                for (i in 1:fact.nrows){
                  sumsq <- 0
                  for (j in seq(1,dim(fact)[2],2)) {
                    sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                  }
                  sumsq
                  f7=f7+function57k0(m,n,d,dh,r1,t,s1, sumsq)
                }
              }
            }
          }
          #sumx
          if (dh==0)
            f7=f7+function57k0(m,n,d,dh,r1,t,s1,0)
        }
        f7
      }
    }
  }
}


#-------------------------------------------------------------------------------
#Name of function: function f8
#-------------------------------------------------------------------------------

for (m in 6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-2,n-d-2),2)){
        #we consider k=1 otherwise max(min(m-p-k+1,n-p),1)
        f7=0
        for (dh in max(0,2*d-m+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
```

```
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]

                for (i in 1:fact.nrows){
                  sumsq <- 0
                  for (j in seq(1,dim(fact)[2],2)) {
                    sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                  }
                  sumsq
                  f8=f8+function58k0(m,n,d,dh,r1,t,s1, sumsq)
                }
              }
            }
          }
          #sumx
          if (dh==0)
            f8=f8+function58k0(m,n,d,dh,r1,t,s1,0)
        }
        f8
      }
    }
  }
}

fk00=f1+f2+f3+f4+f5+f6+f7+f8

#-----------------------------------------------------------------------------
#Name of operation: plot f(k0)
#-----------------------------------------------------------------------------

plot(t,fk00,type="l",lty=2,lwd=1,xlim=c(0.2,4), ylab="fk0(t)",col="red",
font.main=1,col.main="red")
```

A.3 R Codes for Estimating IBP Density Function Under $(k^{'}, k)$ Control Policy

```
#------------------------------------------------------------------------------#
              #Name of function: Chapter 5 density function#
                  #Aim: Graphing the density function#
#------------------------------------------------------------------------------#


#-------------------------------------------------------------------------------
#Name of function: f1k0
#Aim: to calculate f1k0(t)
#Input:lambda1, beta1, mu1, sumf1k, k0, kk and k
#Output: the calculation of f1k0(t)
#-------------------------------------------------------------------------------

lambda1=5
beta1=0.6
mu1=4
sumf1k=0
kk=1
k0=2
k=3

t <- as.matrix(seq(0.2,4,0.01))
for (m in 5:10){
  for (ks in 0:(k-kk-1)) {
    for (n in 2:(m-ks)){
       for (s1 in 1: floor(((n-k0)/(k-kk))+1)){
        if (((2*n)-k0-((s1-1)*(k-kk))) < 1) {
          sumf1k<-0
          break
        }


        a11=exp((-lambda1-mu1)*t)
        a12=(k0+(s1-1)*(k-kk))
        a13=(2*n-k0-(s1-1)*(k-kk))
        a14=choose((2*n-k0-((s1-1)*(k-kk))),n)
        a15=lambda1^(m-k0)
        a16=(beta1^n)/factorial(ks+(s1-1)*(k-kk))
        a1=a11*(a12/a13)*a14*a15*a16

          sum21=0
          for (i in 0:30) {
            if ((((s1-1)*(k-kk))+ks+i) < 0) {
              sum21<-0
              break
            }
            if ((m+n-k0+i) < 0) {
              sum21<-0
```

```
              break
            }

            f31=((mu1)^(n+i))/factorial(i)
            f32=t^(m+n-k0+i)
            f33=factorial(((s1-1)*(k-kk))+ks+i)/factorial(m+n-k0+i)
            sum21=sum21+(f31*f32*f33)
          }
      sum21
      sumf1k=sumf1k+(a1*sum21)
      }
    }
  }
}

f1=sumf1k


#------------------------------------------------------------------------------
#Name of function: f2k0
#Aim: to calculate f2k0(t)
#Input:lambda1, beta1, mu1, sumf2k, k0, kk and k
#Output: the calculation of f2k0(t)
#------------------------------------------------------------------------------

lambda1=5
beta1=0.6
mu1=4
sumf2k=0
kk=1
k0=2
k=3


t <- as.matrix(seq(0.2,4,0.01))
for (m in 5:10){
  for (n in 2:(m-1)){
    for (s1 in 1: min(floor(((m-k0)/(k-kk))),floor(((n-k0+kk)/(k-kk))))){
      if ((m-k0-s1*(k-kk)) < 0) {
        sum11<-0
        break
        }

      if ((n-k0-s1*(k-kk)-kk) < 0) {
        sum11<-0
        break
        }

      a11=exp((-lambda1-mu1)*t)
      a12=choose((m+n-(s1*(k-kk))-k0),n)-choose((m+n-(s1*(k-kk))-k0),m-kk)
      a13=lambda1^(m-k0)
```

```
      a14=(beta1^n)/factorial(s1*(k-kk))
      a1=a11*a12*a13*a14

        sum21=0
        for (i in 0:50) {
          if ((m+n-k0+i+1) < 1) {
              sum21<-0
              break
          }

          f31=((mu1)^(n+i))/factorial(i)
          f32=t^(m+n-k0+i)
          f33=factorial((s1*(k-kk))+i)/factorial(m+n-k0+i)
          sum21=sum21+(f31*f32*f33)
        }
    sum21
    sumf2k=sumf2k+(a1*sum21)
    }
  }
}

f2=sumf2k


#-----------------------------------------------------------------------------
#Name of function: factMatrix
#Aim: to get all possible pairs of l's and L's based on the possible values of
#l's and L's
#Input: m,n,r1
#Output: all possible pairs of l's and L's based on the possible values of
#l's and L's
#-----------------------------------------------------------------------------

factMatrix<- function(m,n,r1) {
    library(AlgDesign)
    lengths <-rep(0,2*r1)
    first <-rep(0,2*r1)
    last <- rep(0,2*r1)
    lengths[1] <- m-1
    first[1] <- 2
    last[1] <- m
    lengths[2*r1] <- n
    first[2*r1] <- 1
    last[2*r1] <- n

    if (r1 > 1) {
        for (i in 2:r1) {
            lengths[i] <- m
            first[i] <- 1
            last[i] <- m
        }
```

```
    }
    if (r1 > 1 ) {
        for (i in (r1+1):(2*r1-1)) {
          lengths[i] <- n
          first[i] <- 1
          last[i] <- n

        }
    }

    #generate design matrix
    design <- gen.factorial(lengths)
    design <- design - max(design +1)

    #recode design matrix
    for (i in 1:(2*r1)) {
        #extract the unique values in column i
        u <- sort(unique(design[,i]),decreasing = TRUE)
        j<-0
        for (l in seq(last[i],first[i],-1)) {
            j <- j+1
            design[design[,i] == u[j],i] <- l
        }
    }
    design
}


#-------------------------------------------------------------------------------
#Name of function: function2
#Aim: getting pairs of l's and L's  satisfying the condition
#Input: design,m,n,d,r1,s1
#Output:all possible pairs of l's and L's satisfying the condition
#-------------------------------------------------------------------------------

function2<-function(design,m,n,d,r1,s1){
    count.no.rows=0
    DSLatticePath<-design
    for(i in 1:dim(design)[1]) {
        cond=TRUE
        sum1=0
        sum2=0
        for (j in 1:r1) {
          sum1<-sum1+design[i,j]
          sum2<-sum2+design[i,j+r1]
          cond<-(cond&&(sum1>=sum2))
        }
        cond<-(cond&&(sum1==m-d)&&(sum2==n-d))
        if (cond) {
            count.no.rows=count.no.rows+1
            DSLatticePath[count.no.rows,]<-design[i,]
```

```
        }
    }
    DSLatticePath[1:count.no.rows,]
}



#------------------------------------------------------------------------------
#Name of function: function3
#Aim: finding touch and sequence of touchs
#Input: DSLatticePath,r1
#Output: touch and sequence of touchs
#------------------------------------------------------------------------------

function3<-function(DSLatticePath,r1){
    numberofpaths= dim(DSLatticePath)[1]
    count.touch=matrix(c(rep(0,r1*numberofpaths)), nrow=numberofpaths, ncol=r1)
    for(i in 1:numberofpaths) {
        sum11=0
        sum22=0
        for (j in 1:r1) {
            sum11<-sum11+DSLatticePath[i,j]
            sum22<-sum22+DSLatticePath[i,j+r1]
            if (sum11==sum22) {
                touch<-1
            }
            else {
                touch<-0
            }
             count.touch[i,j]<-touch
        }
    }
  cbind(DSLatticePath,count.touch)
}


#-----------------------------------------------------------------------------
#Name of function: function4
#Aim: to find the position of touching (u_vector)
#Input: DSLatticePath,r1, only rows with a touch or a fealisable lattice path
#       with touch sequence
#Output: u_vector : vector of the position of touching
#-----------------------------------------------------------------------------

function4<-function(arowoffunction3, r1){
    low<-2*r1+1
    high<-3*r1

    ## row getting the touch sequence
    temp<-arowoffunction3
    touchsequence<-(temp[low:high])
```

```
    ##counting touches
    no.of.touch<-sum(temp[low:high])
    touchingpositionvector<-rep(0,no.of.touch+1)
    ##generating ui sequences
    touchingpositionvector[1]<-1

    if (no.of.touch > 0 ) {
        touchingpositionvector[2:(no.of.touch+1)]<-which(touchsequence==1) + 1
        touchingpositionvector
    }
}


#-----------------------------------------------------------------------------
#Name of function: various function5 i.e. function f53k0, f54k0, ..., f57k0
#Aim: to create the R program for various function5.
#The first function5 is f3k0(t)
#Input: m,n,d,dh,r1,t,s1,sumsq
#Output: the function of the summation of f3k0(t)
#-----------------------------------------------------------------------------

function53k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1,d-dh)
    f13=(alpha1^d)*(beta1^(n-2*d))*(lambda1^(m-d-k0))*(mu1^(n-d))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*(k-kk))-sumsq < 1) {
                sum2<-0
                break
            }
            if (s1*(k-kk)< 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*(k-kk))+i)/factorial(s1*(k-kk))
            f35=factorial(m+n-(2*d)-k0-(s1*(k-kk))+j-sumsq-1)
            f36=factorial(m+n-(2*d)-k0-(s1*(k-kk))-sumsq-1)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f53k0=f1*sum2
}
```

```
#----------------------------------------------------------------------------
#Name of function: function f54k0
#----------------------------------------------------------------------------

function54k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-2*d))*(lambda1^(m-d-k0))*(mu1^(n-d))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if ((m+n-(2*d)-k0-s1*(k-kk)-sumsq) < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k+i+j)
            f34=factorial((s1*(k-kk))+i)/factorial(s1*(k-kk)-1)
            f35=factorial(m+n-(2*d)-k0-s1*(k-kk)+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-s1*(k-kk)-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f54k0=f1*sum2
}


#----------------------------------------------------------------------------
#Name of function: function f55k0
#----------------------------------------------------------------------------

function55k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-2*d))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if ((m+n-(2*d)-k0-((s1-1)*k)-ks-sumsq-1) < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i-1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial(((s1-1)*k)+ks+i)/factorial(((s1-1)*k)+ks-1)
            f35=factorial(m+n-(2*d)-k0-((s1-1)*k)-ks+j-sumsq-1)
```

```
            f36=factorial(m+n-(2*d)-k0-((s1-1)*k)-ks-sumsq-1)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f55k0=f1*sum2
}


#-------------------------------------------------------------------------------
#Name of function: function f56k0
#-------------------------------------------------------------------------------

function56k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-(2*d)+1))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*(k-kk))-sumsq < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i-1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial(((s1-1)*(k-kk))+i)/factorial(((s1-1)*(k-kk))-1)
            f35=factorial(m+n-(2*d)-k0-(s1*(k-kk))+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-(s1*(k-kk))-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f56k0=f1*sum2
}

#-------------------------------------------------------------------------------
#Name of function: function f57k0
#-------------------------------------------------------------------------------

function57k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-(2*d)+1))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*(k-kk))-sumsq < 0) {
```

```
                sum2<-0
                break
            }
            f31=(mu2)^(d+i-1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*(k-kk))+i)/factorial((s1*(k-kk))-1)
            f35=factorial(m+n-(2*d)-k0-(s1*(k-kk))+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-(s1*(k-kk))-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f57k0=f1*sum2
}

#--------------------------------------------------------------------------------
#Name of function: function f58k0
#--------------------------------------------------------------------------------

function58k0<-function(m,n,d,dh,r1,t,s1,sumsq) {
    f11=exp((-lambda1-mu2)*t)
    f12=choose(n-d-r1+1,d-dh)
    f13=(alpha1^d)*(beta1^(n-(2*d)+1))*(lambda1^(m-d-k0))*(mu1^(n-d+1))
    f1=f11*f12*f13
    sum2=0
    for (i in 0:50) {
        for (j in 0:50) {
            if (m+n-(2*d)-k0-(s1*(k-kk))-sumsq < 0) {
                sum2<-0
                break
            }
            f31=(mu2)^(d+i-1)/factorial(i)
            f32=(((mu2-mu1)^j)/factorial(j))
            f33=t^(m+n-d-k0+i+j)/factorial(m+n-d-k0+i+j)
            f34=factorial((s1*(k-kk))+i)/factorial((s1*(k-kk))-1)
            f35=factorial(m+n-(2*d)-k0-(s1*(k-kk))+j-sumsq)
            f36=factorial(m+n-(2*d)-k0-(s1*(k-kk))-sumsq)
            sum2=sum2+(f31*f32*f33*f34*(f35/f36))
        }
        sum2
    }
    f58k0=f1*sum2
}


#--------------------------------------------------------------------------------
#Name of function: function6
#Aim: to get the design of R8
#Input: lis
#Output: the calculation of (l(is)-p(is))
```

```
#-------------------------------------------------------------------------------

function6<-function(lis){
    len.lis <- length(lis)
    gen.fact <- paste("gen.factorial(c(1,",lis[1]+1)
    if (len.lis >= 2) {
        for (i in 2:len.lis) {
            gen.fact <- paste(gen.fact,",1,",lis[i]+1)
        }
    }

    gen.fact <- paste(gen.fact,"))")
    #columns of fact are l1s, p1s, l2s, p2s, l3s, p3s, ...
    fact <- eval(parse(text=gen.fact))
    fact <- fact - max(fact + 1)

    #coding given in terms -1 0 1 etc so need to recode
    fact.ncols <- dim(fact)[2]
    j<-1
    for (i in 1:len.lis) {
        #relabel
        fact[,j] <- lis[i]
        j <- j+1
        #extract unique values in column i
        u <- sort(unique(fact[,j]),decreasing=TRUE)
        if (i == 1) {
            for (l in seq(1,lis[i])) {
                fact[fact[,j] == u[l],j] <- l
            }
        }
        else {
             for (l in seq(0,lis[i])) {
                fact[fact[,j] == u[l+1],j] <- l
             }
        }
        fact[lis[i]==0,j] <- 0
        j <- j+1
    }
    fact
  }



#-------------------------------------------------------------------------------
#Input:lambda1,alpha1,beta1,mu1, mu2, kk, k0, k
#Output: the calculation of f3k0(t), f4k0(t), f5k0(t), f6k0(t), f7k0(t)
#-------------------------------------------------------------------------------

t <- as.matrix(seq(0.2,4,0.01))
lambda1=5
alpha1=0.4
```

```
beta1=0.6
mu1=8
mu2=4
kk=1
k0=2
k=3
#-------------------------------------------------------------------------------
#Name of function: f3
#-------------------------------------------------------------------------------

for (m in 5:6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-kk,n-d+1),2)){
        f3=0
        for (dh in max(0,2*d-n+r1-1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/(k-kk))) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= (k-kk)) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - (k-kk)
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]

                for (i in 1:fact.nrows){
                  sumsq <- 0
```

```
                for (j in seq(1,dim(fact)[2],2)) {
                  sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                }
                sumsq
                f3=f3+function53k0(m,n,d,dh,r1,t,s1, sumsq)
              }
            }
          }
        }
        #sumx
        if (dh==0)
          f3=f3+function53k0(m,n,d,dh,r1,t,s1,0)
      }
      f3
    }
  }
}
}
```

```
#--------------------------------------------------------------------------------
#Name of function: function f4
#--------------------------------------------------------------------------------

for (m in 5:6){
  for (n in 2:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-kk,n-d+1),2)){
        f4=0
        for (dh in max(0,2*d-n+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
```

```
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]

                for (i in 1:fact.nrows){
                  sumsq <- 0
                  for (j in seq(1,dim(fact)[2],2)) {
                    sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                  }
                  sumsq
                  f4=f4+function54k0(m,n,d,dh,r1,t,s1, sumsq)
                }
              }
            }
          }
          #sumx
          if (dh==0)
            f4=f4+function54k0(m,n,d,dh,r1,t,s1,0)
        }
       f4
      }
    }
  }
}


#-------------------------------------------------------------------------------
#Name of function: function f5
#-------------------------------------------------------------------------------

for (m in 7:7){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-kk,n-d+1),2)){
        f5=0
        for (dh in max(0,2*d-n+r1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
```

```
            ui.vector <- function4(DSLatticePath2[rowDS,],r1)
            if (length(ui.vector)== s1+1) {
              for (j in 2: (length(ui.vector))) {
                if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                  DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                  ui.vector[j]] - 2
                }
              }
            }
          }
        }
        count.no.touch<-function3(DSLatticePath,r1)
        #Calculation of the summation
        if (dh != 0) {
          num.rows<-dim(DSLatticePath)[1]
          cmb<-combn(c(1:r1),dh)
          for (r in 1:num.rows){
            for (g in 1:dim(cmb)[2]){
              indices<-cmb[,g]
              lis<-DSLatticePath[r,indices]
              is.numeric(lis)
              if (length(lis)> 1) lis<-as.numeric(lis)

              #print(c("len.lis=", len.lis))
              fact <- function6(lis)
              #get the sum
              fact.nrows <- dim(fact)[1]

              for (i in 1:fact.nrows){
                sumsq <- 0
                for (j in seq(1,dim(fact)[2],2)) {
                  sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                }
                sumsq
                f5=f5+function55k0(m,n,d,dh,r1,t,s1, sumsq)
              }
            }
          }
        }
        #sumx
        if (dh==0)
          f5=f5+function55k0(m,n,d,dh,r1,t,s1,0)
      }
      f5
    }
  }
 }
}


#-------------------------------------------------------------------------------
```

```
#Name of function: function f6
#--------------------------------------------------------------------------------

for (m in 6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-kk,n-d+1),2)){
        #we consider k=1 otherwise max(min(m-p-k+1,n-p),1)
        f6=0
        for (dh in max(0,2*d-n+r1-1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]

                for (i in 1:fact.nrows){
                  sumsq <- 0
                  for (j in seq(1,dim(fact)[2],2)) {
                    sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                  }
                  sumsq
                  f6=f6+function56k0(m,n,d,dh,r1,t,s1, sumsq)
```

```
              }
            }
          }
        }
        #sumx
        if (dh==0)
          f6=f6+function56k0(m,n,d,dh,r1,t,s1,0)
      }
      f6
    }
  }
 }
}
```

```
#--------------------------------------------------------------------------------
#Name of function: function f7
#--------------------------------------------------------------------------------

for (m in 6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-kk,n-d+1),2)){
        #we consider k=1 otherwise max(min(m-p-k+1,n-p),1)
        f7=0
        for (dh in max(0,2*d-n+r1-1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
```

```
              indices<-cmb[,g]
              lis<-DSLatticePath[r,indices]
              is.numeric(lis)
              if (length(lis)> 1) lis<-as.numeric(lis)

              #print(c("len.lis=", len.lis))
              fact <- function6(lis)
              #get the sum
              fact.nrows <- dim(fact)[1]

              for (i in 1:fact.nrows){
                sumsq <- 0
                for (j in seq(1,dim(fact)[2],2)) {
                  sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                }
                sumsq
                f7=f7+function57k0(m,n,d,dh,r1,t,s1, sumsq)
              }
            }
          }
        }
        #sumx
        if (dh==0)
          f7=f7+function57k0(m,n,d,dh,r1,t,s1,0)
      }
      f7
    }
  }
 }
}



#------------------------------------------------------------------------------
#Name of function: function f8
#------------------------------------------------------------------------------

for (m in 6){
  for (n in 3:(m-1)){
    for (d in 1:min(floor(n/2),(m-2))){
      for (r1 in 2:max(min(m-d-kk,n-d+1),2)){
        #we consider k=1 otherwise max(min(m-p-k+1,n-p),1)
        f7=0
        for (dh in max(0,2*d-n+r1-1):(min(r1,d))) {
          for (s1 in 1:floor((n-k0)/k)) {
            DSLatticePath<-function2(factMatrix(m,n,r1),m,n,d,r1)
            ####    assume k=1
            DSLatticePath2<-function3(DSLatticePath,r1)
            num.Rows.DSLatticePath2 <- dim(DSLatticePath2)[1]
            for (rowDS in 1:num.Rows.DSLatticePath2) {
              ui.vector <- function4(DSLatticePath2[rowDS,],r1)
```

```
              if (length(ui.vector)== s1+1) {
                for (j in 2: (length(ui.vector))) {
                  if ( DSLatticePath[rowDS, ui.vector[j]] >= 2) {
                    DSLatticePath[rowDS, ui.vector[j]] <- DSLatticePath[rowDS,
                    ui.vector[j]] - 2
                  }
                }
              }
            }
          }
          count.no.touch<-function3(DSLatticePath,r1)
          #Calculation of the summation
          if (dh != 0) {
            num.rows<-dim(DSLatticePath)[1]
            cmb<-combn(c(1:r1),dh)
            for (r in 1:num.rows){
              for (g in 1:dim(cmb)[2]){
                indices<-cmb[,g]
                lis<-DSLatticePath[r,indices]
                is.numeric(lis)
                if (length(lis)> 1) lis<-as.numeric(lis)

                #print(c("len.lis=", len.lis))
                fact <- function6(lis)
                #get the sum
                fact.nrows <- dim(fact)[1]

                for (i in 1:fact.nrows){
                  sumsq <- 0
                  for (j in seq(1,dim(fact)[2],2)) {
                    sumsq <- sumsq + sum(fact[i,j]-fact[i,j+1])
                  }
                  sumsq
                  f8=f8+function58k0(m,n,d,dh,r1,t,s1, sumsq)
                }
              }
            }
          }
          #sumx
          if (dh==0)
            f8=f8+function58k0(m,n,d,dh,r1,t,s1,0)
        }
        f8
      }
    }
  }
} fk00=f1+f2+f3+f4+f5+f6+f7+f8

#--------------------------------------------------------------------------------
#Name of operation: plot f(k0)
#--------------------------------------------------------------------------------
```

```
plot(t,fk00,type="l",lty=2,lwd=1,xlim=c(0.2,4), ylim=c(0,0.1),ylab="Chp5fk0(t)"
,col="red",
font.main=1,col.main="red")
```

```
#plot(t,fk00,type="l",lty=2,lwd=1,ylab=c(expression(f[k](t))),col="red",
#xlim=c(0,1.5),ylim=c(0,0.5), font.main=1,col.main="red")
#lines(t,fk01,lty=3,lwd=1,col="blue")
#lines(t,fk02,lty=4,lwd=1,col="black")
#legend(x=0.75,y=4.0, c(expression(paste(alpha==0.2,", ",beta==0.8)),
#expression(paste(alpha==0.4,", ",beta==0.6)),expression(paste(alpha==0.8,
#", ",beta==0.2))), cex=0.8,lty=c(2,3,4),col=c("red","blue","black"),lwd=1);
```

Appendix B

Derivation of Pdf of the Total Vacation Period of IBP of $M/C_2/1$ Queueing System

B.1 Derivation of Pdf of the Total Vacation Period of IBP of $M/C_2/1$ Under $(0, k)$ Control Policy and Ending in Vacation Period

To derive the solutions which have been found in chapter 4, we require to solve the multiple integration below.

$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}\lambda^k \tau_y^{k-1}}{\Gamma(k)} \frac{e^{-\lambda\tau_s}\lambda^{k^*}\tau_s^{k^*-1}}{\Gamma(k^*)} d\tau_s d\tau_y = \frac{e^{-\lambda\tau}\lambda^{(s-1)k}\lambda^{k^*}}{(\Gamma(k))^{s-1}\Gamma(k^*)} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \tau_y^{k-1}\tau_s^{k^*-1} d\tau_s d\tau_y$$

First integrate over τ_s, where $\tau_s = \tau - \tau_1 + \tau_2 + \ldots + \tau_{s-1}$.

$$\begin{aligned}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}} \tau_s^{k^*-1} d\tau_s &= \frac{\tau_s^{k^*}}{k^*}\Big|^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}} \\ &= \frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k^*}}{k^*}\end{aligned}$$

Next for integration over τ_{s-1}, we calculate second integration as follows.

$$\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}} \frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k^*}}{k^*}\tau_{s-1}^{k-1}d\tau_{s-1}$$

$$= \frac{1}{k^*}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k^*}$$

$$\left(1-\frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k^*}\tau_{s-1}^{k-1}d\tau_{s-1}$$

$$= \frac{1}{k^*}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k^*}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}}$$

$$\left(1-\frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k^*}\tau_{s-1}^{k-1}d\tau_{s-1}$$

Let $u=\frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}$, $d\tau_{s-1}=(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2})du$

Then the above right hand side equation becomes

$$\frac{1}{k^*}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k+k^*}\int_0^1(1-u)^{k^*}u^{k-1}du$$

$$= \frac{1}{k^*}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k+k^*}\frac{\Gamma(k^*+1)\Gamma(k)}{\Gamma(k+k^*+1)}$$

$$= \frac{\Gamma(k^*)\Gamma(k)}{\Gamma(k+k^*+1)}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k+k^*}$$

We need to continue to calculate the integration over $\tau_{s-2},\tau_{s-3},\ldots,\tau_2,\tau_1$.

For integration over τ_1, we calculate

$$\int_0^{\tau}\frac{\Gamma(k^*)(\Gamma(k))^{s-2}}{\Gamma((s-2)k+k^*+1)}(\tau-\tau_1)^{(s-2)k+k^*}\tau_1^{k-1}d\tau_1 = \frac{\Gamma(k^*)(\Gamma(k))^{s-2}}{\Gamma((s-2)k+k^*+1)}$$

$$\int_0^{\tau}\tau^{(s-2)k+k^*}\left(1-\frac{\tau_1}{\tau}\right)^{k^*+(s-2)k}\tau_1^{k-1}d\tau_1$$

Let $u=\frac{\tau_1}{\tau}$, $d\tau_1=\tau du$

Then the above right hand side equation becomes

$$= \frac{\Gamma(k^*)(\Gamma(k))^{s-2}}{\Gamma(k^*+(s-2)k+1)}\int_0^1\tau^{k^*+(s-2)k}(1-u)^{k^*+(s-2)k}(u\tau)^{k-1}\tau du$$

$$= \frac{\Gamma(k^*)(\Gamma(k))^{s-2}}{\Gamma(k^*+(s-2)k+1)}\tau^{k^*+(s-1)k}\int_0^1(1-u)^{k^*+(s-2)k}u^{k-1}\tau du$$

$$= \frac{\Gamma(k^*)(\Gamma(k))^{s-2}}{\Gamma(k^*+(s-2)k+1)}\tau^{k^*+(s-1)k}\frac{\Gamma(k^*+(s-2)k)\Gamma(k)}{\Gamma(k^*+(s-1)k+1)}$$

$$= \frac{\Gamma(k^*)(\Gamma(k))^{s-1}}{\Gamma(k^*+(s-1)k+1)}\tau^{k^*+(s-1)k}$$

Finally we get

$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}(\lambda\tau_y)^{k-1}}{\Gamma(k)} \frac{e^{-\lambda\tau_s}(\lambda\tau_s)^{k^*-1}}{\Gamma(k^*)} d\tau_s d\tau_y = \frac{e^{-\lambda\tau}\lambda^{(s-1)k}\lambda^{k^*}}{(\Gamma(k))^{s-1}\Gamma(k^*)}$$
$$\frac{\Gamma(k^*)(\Gamma(k))^{s-1}}{\Gamma(k^* + (s-1)k + 1)}\tau^{k^*+(s-1)k}.$$

which yields

$$f_{s(kk^*)}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{(s-1)k+k^*}}{\Gamma((s-1)k + k^* + 1)}.$$

B.2 The Integration of the Integral Part of $f^1_{k_0;k}(t)$

For the integral part of $f^1_{k_0;k}(t)$, we find

$$\begin{aligned}
&\int_{\tau=0}^{t} \frac{e^{-\lambda\tau}(\lambda\tau)^{(s-1)k+k^*}}{\Gamma((s-1)k+k^*+1)} \\
&\times \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{m+n-(s-1)k-k^*-k_0}(t-\tau)^{m+n-(s-1)k-k^*-k_0-1}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
&\times \left(\frac{\lambda}{\lambda+\mu_1}\right)^{m-(s-1)k-k^*-k_0} \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n} d\tau \\
&= \lambda^{m-k_0}\frac{(\beta\mu_1)^n}{\Gamma(k^*+(s-1)k+1)}\frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
&\times \int_{\tau=0}^{t} e^{\mu_1\tau}\tau^{k^*+(s-1)k}(t-\tau)^{m+n-(s-1)k-k^*-k_0-1}d\tau \\
&= \lambda^{m-k_0}\frac{\beta^n}{\Gamma(k^*+(s-1)k+1)}\frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
&\times \int_{\tau=0}^{t}\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}\tau^{k^*+(s-1)k+i}(t-\tau)^{m+n-(s-1)k-k^*-k_0-1}d\tau.
\end{aligned}$$

Let $\upsilon = \frac{\tau}{t}$ then the right hand side expression becomes

$$\begin{aligned}
&= \lambda^{m-k_0}\frac{\beta^n}{\Gamma((s-1)k+k^*+1)}\frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
&\times \sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}\int_{\tau=0}^{1}(\upsilon t)^{k^*+(s-1)k+i}(t-\upsilon)^{m+n-(s-1)k-k^*-k_0-1}t d\upsilon \\
&= \lambda^{m-k_0}\frac{\beta^n}{\Gamma((s-1)k+k^*+1)}\frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
&\times \sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}t^{m+n-k_0+i}\int_{\tau=0}^{1}\upsilon^{k^*+(s-1)k+i}(1-\upsilon)^{m+n-(s-1)k-k^*-k_0-1}d\upsilon.
\end{aligned}$$

The above right hand side expression can be simplified into

$$\begin{aligned}
&\lambda^{m-k_0}\frac{\beta^n}{\Gamma((s-1)k+k^*+1)}\frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-(s-1)k-k^*-k_0)} \\
&\times \sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}t^{m+n-k_0+i}\frac{\Gamma(k^*+(s-1)k+i+1)\Gamma(m+n-(s-1)k-k^*-k_0)}{\Gamma(m+n-k_0+i+1)} \\
&= \lambda^{m-k_0}\frac{\beta^n e^{-(\lambda+\mu_1)t}}{\Gamma((s-1)k+k^*+1)}\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}t^{m+n-k_0+i}\frac{\Gamma((s-1)k+k^*+i+1)}{\Gamma(m+n-k_0+i+1)}.
\end{aligned} \tag{B.1}$$

B.3 Derivation of Pdf of the Total Vacation Period of IBP of $M/C_2/1$ under $(0,k)$ Control Policy and Ending in Busy Period

$$\int_{\tau_1}\cdots\int_{\tau_s}\prod_{y=1}^{s}\frac{e^{-\lambda\tau_y}(\lambda\tau_y)^{k-1}}{\Gamma(k)}d\tau_y=\frac{e^{-\lambda\tau}\lambda^{sk}}{(\Gamma(k))^s}\int_{\tau_1}\cdots\int_{\tau_s}\prod_{y=1}^{s}\tau_y^{k-1}d\tau_y.$$

First integrate over τ_s, where $\tau_s=\tau-\tau_1+\tau_2+\ldots+\tau_{s-1}$.

$$\begin{aligned}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}}\tau_s^{k-k'-1}d\tau_s &= \frac{\tau_s^{k-k'}}{k-k'}|^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}}\\ &= \frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k-k'}}{k-k'}\end{aligned}$$

Next for integration over τ_{s-1}, we calculate second integration as follows.

$$\begin{aligned}&\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}}\frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k-k'}}{k-k'}\tau_{s-1}^{k-k'-1}d\tau_{s-1}\\ &=\frac{1}{k-k'}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'}\\ &\left(1-\frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k-k'}\tau_{s-1}^{k-k'-1}d\tau_{s-1}\\ &=\frac{1}{k-k'}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}}\\ &\left(1-\frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k-k'}\tau_{s-1}^{k-k'-1}d\tau_{s-1}\end{aligned}$$

Let $u=\frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}$, $d\tau_{s-1}=(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2})du$

Then the above right hand side equation becomes

$$\begin{aligned}&\frac{1}{k-k'}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{2(k-k')}\int_0^1(1-u)^{k-k'}u^{k-k'-1}du\\ &=\frac{1}{k-k'}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{2(k-k')}\frac{\Gamma(k-k'+1)\Gamma(k-k')}{\Gamma(2(k-k')+1)}\\ &=\frac{(\Gamma(k-k'))^2}{\Gamma(2(k-k')+1)}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{2(k-k')}\end{aligned}$$

We need to continue to calculate the integration over $\tau_{s-2},\tau_{s-3},\ldots,\tau_2,\tau_1$.

For integration over τ_1, we calculate

$$\begin{aligned}&\int_0^{\tau}\frac{(\Gamma(k-k'))^{s-1}}{\Gamma((s-1)(k-k')+1)}(\tau-\tau_1)^{(s-1)(k-k')}\tau_1^{k-k'-1}d\tau_1\\ &=\frac{(\Gamma(k-k'))^{s-1}}{\Gamma((s-1)(k-k')+1)}\int_0^{\tau}\tau^{(s-1)(k-k')}\left(1-\frac{\tau_1}{\tau}\right)^{(s-1)(k-k')}\tau_1^{k-k'-1}d\tau_1\end{aligned}$$

Let $u = \frac{\tau_1}{\tau},\ d\tau_1 = \tau du$

Then the above right hand side equation becomes

$$\begin{aligned}
&= \frac{(\Gamma(k))^{s-1}}{\Gamma((s-1)k+1)} \int_0^\tau \tau^{(s-1)k}(1-u)^{(s-1)k}(u\tau)^{k-1}\tau du \\
&= \frac{(\Gamma(k))^{s-1}}{\Gamma((s-1)k+1)} \tau^{sk} \int_0^1 (1-u)^{(s-1)k}u^{k-1}du \\
&= \frac{(\Gamma(k))^{s-1}}{\Gamma((s-1)k+1)} \tau^{sk} \frac{\Gamma((s-1)k+1)\Gamma(k)}{\Gamma(sk+1)} \\
&= \frac{(\Gamma(k))^{s}}{\Gamma(sk+1)} \tau^{sk}
\end{aligned}$$

Finally we get

$$\begin{aligned}
\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s} \frac{e^{-\lambda\tau_y}(\lambda\tau_y)^{k-1}}{\Gamma(k)} d\tau_y &= \frac{e^{-\lambda\tau}\lambda^{s(k-1)}}{(\Gamma(k))^s} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s} \tau_y^{k-1} d\tau_y \\
&= \frac{e^{-\lambda\tau}\lambda^{sk)}}{(\Gamma(k))^s} \frac{(\Gamma(k))^s\tau^{sk}}{\Gamma(sk+1)}
\end{aligned}$$

which yields

$$f_{sk}(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^{sk}}{\Gamma(sk+1)}$$

B.4 The Integration of the Integral Part of $f^2_{k_0;k}(t)$

For the integral part of $f^2_{k_0;k}(t)$, we find

$$\int_{\tau=0}^{t} \frac{e^{-\lambda\tau}\lambda^{s(k-1)}\tau^{sk}}{\Gamma(sk+1)} \frac{e^{-(\lambda+\mu_1)(t-\tau)}(\lambda+\mu_1)^{m+n-k_0-sk}(t-\tau)^{m+n-k_0-sk-1}}{\Gamma(m+n-k_0-sk)}$$
$$\times \left(\frac{\lambda}{\lambda+\mu_1}\right)^{m-k_0-sk} \left(\frac{\beta\mu_1}{\lambda+\mu_1}\right)^{n} d\tau$$
$$= e^{-(\lambda+\mu_1)t} \frac{(\beta\mu_1)^n}{\Gamma(sk+1)} \frac{\lambda^{m-k_0}}{\Gamma(m+n-k_0-sk)} \int_{\tau=0}^{t} e^{\mu_1\tau}\tau^{sk}(t-\tau)^{m+n-k_0-sk-1} d\tau$$
$$= \lambda^{m-k_0} \frac{\beta^n}{\Gamma(sk+1)} \frac{e^{-(\lambda+\mu_1)t}}{\Gamma(m+n-k_0-sk)} \int_{\tau=0}^{t} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} \tau^{sk+i}(t-\tau)^{m+n-k_0-sk-1} d\tau.$$

Let $\upsilon = \frac{\tau}{t}$ then the right hand side expression becomes

$$= e^{-(\lambda+\mu_1)t} \frac{\beta^n}{\Gamma(sk+1)} \frac{\lambda^{m-k_0}}{\Gamma(m+n-k_0-sk)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} \int_{\tau=0}^{1} (\upsilon t)^{sk+i}(t-\upsilon)^{m+n-k_0-sk-1} t d\upsilon$$
$$= e^{-(\lambda+\mu_1)t} \frac{\beta^n}{\Gamma(sk+1)} \frac{\lambda^{m-k_0}}{\Gamma(m+n-k_0-sk)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i}$$
$$\times \int_{\tau=0}^{1} \upsilon^{sk+i}(1-\upsilon)^{m+n-k_0-sk-1} d\upsilon$$
$$= e^{-(\lambda+\mu_1)t} \frac{\beta^n}{\Gamma(sk+1)} \frac{\lambda^{m-k_0}}{\Gamma(m+n-k_0-sk)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i}$$
$$\times \beta(sk+i+1, m+n-k_0-sk).$$

Simplifying the above equation, yields

$$e^{-(\lambda+\mu_1)t} \frac{\beta^n}{\Gamma(sk+1)} \frac{\lambda^{m-k_0}}{\Gamma(m+n-k_0-sk)} \sum_{i=0}^{\infty} \frac{\mu_1^{n+i}}{i!} t^{m+n-k_0+i}$$
$$\times \frac{\Gamma(sk+i+1)\Gamma(m+n-k_0-sk)}{\Gamma(m+n-k_0+i+1)} \tag{B.2}$$

B.5 The Derivation of $f^3_{k_0;k}(t)$

The derivation of $f^3_{k_0;k}(t)$ is given below.

$$
\begin{aligned}
f^3_{k_0;k}(t) = & \sum_{(R_1^1,R_2^1,R_3,R_4,R_5,R_6,R_7,R_8,R_9)} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \\
& \times \int_0^t \int_0^{t-\tau} f_{sk}(\tau) f_{N_1}(t_1) \left(\frac{\lambda}{\lambda+\mu_1}\right)^{N_1-n+d} \left(\frac{\alpha_1\mu_1}{\lambda+\mu_1}\right)^{d} \left(\frac{\beta+\mu_1}{\lambda+\mu_1}\right)^{n-2d} \\
& \times f_{N_2}(t-\tau-t_1) \left(\frac{\lambda}{\lambda+\mu_2}\right)^{N_2-d} \left(\frac{\mu_2}{\lambda+\mu_2}\right)^{d} dt_1 d\tau \\
= & \sum_{R_1^1} \sum_{(R_2^1,R_1^*,R_2^1,R_3,R_4,R_5^1,R_6^1,R_7^1,R_8^1,R_{9^1})} \begin{pmatrix} n-d-r \\ d-d_h \end{pmatrix} \\
& \times \alpha_1^d \beta^{n-2d} \lambda^{sk-s+N_1+N_2-n} \mu_1^{nd} \mu_2^d \frac{e^{-(\lambda+\mu_2)t}}{\Gamma(sk+1)\Gamma(N_1)\Gamma(N_2)} \\
& \times \int_0^t \int_0^{t-\tau} e^{\mu_2\tau} e^{(\mu_2-\mu_1)t_1} \tau^{sk} t_1^{N_1-1} (t-\tau-t_1)^{N_2-1} dt_1 d\tau.
\end{aligned}
\tag{B.3}
$$

For the last part, i.e. the double integral, we find

$$
\begin{aligned}
& \int_0^t \int_0^{t-\tau} e^{\mu_2\tau} e^{(\mu_2-\mu_1)t_1} \tau^{sk} t_1^{N_1-1} (t-\tau-t_1)^{N_2-1} dt_1 d\tau \\
= & \int_0^t \int_0^{t-\tau} \sum_{i=0}^{\infty} \frac{(\mu_2\tau)^i}{i!} \sum_{j=0}^{\infty} \frac{((\mu_2-\mu_1)t_1)^j}{j!} \tau^{sk} t_1^{N_1-1} (t-\tau-t_1)^{N_2-1} dt_1 d\tau \\
= & \sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \int_0^t \tau^{sk+i} \left(\int_0^{t-\tau} t_1^{N_1+j-1} (t-\tau)^{N_2-1} \left(1-\frac{t_1}{t-\tau}\right)^{N_2-1} dt_1 \right) d\tau.
\end{aligned}
\tag{B.4}
$$

By substituting $u = \frac{t_1}{t-\tau}$ in (B.4), we obtain

$$\begin{aligned}
&\int_0^t \int_0^{t-\tau} e^{\mu_2 \tau} e^{(\mu_2-\mu_1)t_1} \tau^{sk} t_1^{N_1-1} (t-\tau-t_1)^{N_2-1} dt_1 d\tau \\
=&\sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \int_0^t \tau^{sk+i} \int_0^1 (u(t-\tau))^{N_1+j-1} (t-\tau)^{N_2-1} (1-u)^{N_2-1} (t-\tau) du d\tau \\
=&\sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \int_0^t \tau^{sk+i} (t-\tau)^{N_1+N_2+j-1} \left(\int_0^1 u^{N_1+j-1} (1-u)^{N_2-1} du \right) d\tau \\
=&\sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \beta(N_1+j, N_2) \int_0^t \tau^{sk+i} (t-\tau)^{N_1+N_2+j-1} d\tau.
\end{aligned} \tag{B.5}$$

Suppose $\upsilon = \frac{\tau}{t}$, then (B.5) can be written as

$$\begin{aligned}
&\sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \beta(N_1+j, N_2) \int_0^t (\upsilon t)^{sk+i} (t-\upsilon t)^{N_1+N_2+j-1} t d\upsilon \\
&= \sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \beta(N_1+j, N_2) t^{sk+N_1+N_2+i+j} \int_0^t \upsilon^{sk+i} (1-\upsilon)^{N_1+N_2+j-1} d\upsilon \\
&= \sum_{i=0}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=0}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \beta(N_1+j, N_2) \beta(sk+i+1, N_1+N_2+j) t^{sk+N_1+N_2+i+j}
\end{aligned} \tag{B.6}$$

B.6 Derivation of Pdf of the Total Vacation Period of IBP of $M/C_2/1$ under $(k^{'},k)$ Control Policy and Ending in Vacation Period

Suppose $\tau = \tau_1 + \tau_2 + \ldots + \tau_s$, where τ the total time in idle. The probability density function of τ is given by

$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}\lambda^{k-k'}\tau_y^{k-k'-1}}{\Gamma(k-k')} \frac{e^{-\lambda\tau_s}\lambda^{*}\tau_s^{k^*-1}}{\Gamma(k^*)} d\tau_s d\tau_y = \frac{e^{-\lambda\tau}\lambda^{(s-1)(k-k')+k^*}}{(\Gamma(k-k'))^{s-1}\Gamma(k^*)} \times$$
$$\int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \tau_y^{k-k'-1}\tau_s^{k^*-1} d\tau_s d\tau_y$$

The integration can be done step by step.

First integrate over τ_s, where $\tau_s = \tau - \tau_1 + \tau_2 + \ldots + \tau_{s-1}$.

$$\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}} \tau_s^{k^*-1} d\tau_s = \frac{\tau_s^{k^*}}{k^*}|^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}} =$$
$$\frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k^*}}{k^*}$$

Next for integration over τ_{s-1}, we calculate second integration as follows.

$$\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}} \frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k^*}}{k^*} \tau_{s-1}^{k-k'-1} d\tau_{s-1}$$
$$= \frac{1}{k^*}\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k^*}$$
$$\times \left(1 - \frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k^*} \tau_{s-1}^{k-k'-1} d\tau_{s-1}$$
$$= \frac{1}{k^*}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k^*} \int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}}$$
$$\times \left(1 - \frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k^*} \tau_{s-1}^{k-k'-1} d\tau_{s-1}$$

Let $u = \frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}$, $d\tau_{s-1} = (\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2})du$

Then the above right hand side equation becomes

$$\frac{1}{k^*}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'+k^*} \int_0^1 (1-u)^{k^*} u^{k-k'-1} du$$
$$= \frac{1}{k^*}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'+k^*} \frac{\Gamma(k^*+1)\Gamma(k-k')}{\Gamma(k-k'+k^*+1)}$$
$$= \frac{\Gamma(k^*)\Gamma(k-k')}{\Gamma(k-k'+k^*+1)}(\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'+k^*}$$

We need to continue to calculate the integration over $\tau_{s-2}, \tau_{s-3}, \ldots, \tau_2, \tau_1$.

For integration over τ_1, we calculate

$$\int_0^{\tau} \frac{\Gamma(k^*)(\Gamma(k-k'))^{s-2}}{\Gamma((s-2)(k-k')+k^*+1)}(\tau-\tau_1)^{(s-2)(k-k')+k^*}\tau_1^{k-k'-1}d\tau_1$$
$$= \frac{\Gamma(k^*)(\Gamma(k-k'))^{s-2}}{\Gamma((s-2)(k-k')+k^*+1)}\int_0^{\tau} \tau^{(s-2)(k-k')+k^*}\left(1-\frac{\tau_1}{\tau}\right)^{(s-2)(k-k')+k^*}\tau_1^{k-k'-1}d\tau_1$$

Let $u = \frac{\tau_1}{\tau},\ d\tau_1 = \tau du$

Then the above right hand side equation becomes

$$= \frac{\Gamma(k^*)(\Gamma(k-k'))^{s-2}}{\Gamma(k^*+(s-2)(k-k')+1)}\int_0^1 \tau^{k^*+(s-2)(k-k')}(1-u)^{k^*+(s-2)(k-k')}(u\tau)^{k-k'-1}\tau du$$
$$= \frac{\Gamma(k^*)(\Gamma(k-k'))^{s-2}}{\Gamma(k^*+(s-2)(k-k')+1)}\tau^{k^*+(s-1)(k-k')}\int_0^1 (1-u)^{k^*+(s-2)(k-k')}u^{k-k'-1}du$$
$$= \frac{\Gamma(k^*)(\Gamma(k-k'))^{s-2}}{\Gamma(k^*+(s-2)(k-k')+1)}\tau^{k^*+(s-1)(k-k')}\frac{\Gamma(k^*+(s-2)(k-k')+1)\Gamma(k-k')}{\Gamma(k^*+(s-1)(k-k')+1)}$$
$$= \frac{\Gamma(k^*)(\Gamma(k-k'))^{s-1}}{\Gamma(k^*+(s-1)(k-k')+1)}\tau^{k^*+(s-1)(k-k')}$$

Finally we get

$$\int_{\tau_1}\cdots\int_{\tau_s}\prod_{y=1}^{s-1}\frac{e^{-\lambda\tau_y}\lambda^{k-k'}\tau_y^{k-k'-1}}{\Gamma(k-k')}\frac{e^{-\lambda\tau_s}\lambda^*\tau_s^{k^*-1}}{\Gamma(k^*)}d\tau_s d\tau_y = \frac{e^{-\lambda\tau}\lambda^{(s-1)(k-k')+k^*}}{(\Gamma(k-k'))^{s-1}\Gamma(k^*)}\times$$
$$\int_{\tau_1}\cdots\int_{\tau_s}\prod_{y=1}^{s-1}\tau_y^{k-k'-1}\tau_s^{k^*-1}d\tau_s d\tau_y$$
$$= \frac{e^{-\lambda\tau}\lambda^{(s-1)(k-k'-1)+k^*}}{(\Gamma(k-k'))^{s-1}\Gamma(k^*)}\frac{\Gamma(k^*)(\Gamma(k-k'))^{s-1}}{\Gamma(k^*+(s-1)(k-k')+1)}\tau^{k^*+(s-1)(k-k')}$$
$$= \frac{e^{-\lambda\tau}(\lambda\tau)^{k^*+(s-1)(k-k'-1)}}{\Gamma(k^*+(s-1)(k-k')+1)}$$

which yields

$$f_{s(k-k')(\tau)} = \frac{e^{-\lambda\tau}(\lambda\tau)^{k^*+(s-1)(k-k'-1)}}{\Gamma(k^*+(s-1)(k-k')+1)}$$

B.7 The Derivation of the Integral Part of $f^1_{k_0;(k',k)}(t)$

The integral part of $f^1_{k_0;(k',k)}(t)$ is equal to

$$
\begin{aligned}
& e^{-(\lambda+\mu_1)t}\frac{\lambda^{m-k_0}}{\Gamma((s-1)(k-k')+k^*+1)}\frac{e^{(\beta\mu_1)^n}}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\
& \times\int_{\tau=0}^{t} e^{\mu_1\tau}\tau^{k^*+(s-1)(k-k')}(t-\tau)^{m+n-(s-1)(k-k')-k^*-k_0-1}d\tau \\
=& e^{-(\lambda+\mu_1)t}\frac{\lambda^{m-k_0}}{\Gamma(k^*+(s-1)k+1)}\frac{\beta^n}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\
& \times\int_{\tau=0}^{t}\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}\tau^{k^*+(s-1)(k-k')+i}(t-\tau)^{m+n-(s-1)(k-k')-k^*-k_0-1}d\tau.
\end{aligned}
$$

Let $\upsilon=\frac{\tau}{t}$ then the right hand side expression becomes

$$
\begin{aligned}
&= e^{-(\lambda+\mu_1)t}\frac{\lambda^{m-k_0}}{\Gamma(k^*+(s-1)(k-k')+1)}\frac{\beta^n}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\
&\times\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}\int_{\tau=0}^{t}(\upsilon t)^{k^*+(s-1)(k-k')+i}(t-\upsilon t)^{m+n-(s-1)(k-k')-k^*-k_0-1}t\ d\upsilon \\
&= e^{-(\lambda+\mu_1)t}\frac{\lambda^{m-k_0}}{\Gamma(k^*+(s-1)(k-k')+1)}\frac{\beta^n}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\
&\times\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}\int_{\tau=0}^{t}(\upsilon t)^{k^*+(s-1)(k-k')+i}(t-\upsilon t)^{m+n-(s-1)(k-k')-k^*-k_0-1}t\ d\upsilon \\
&= e^{-(\lambda+\mu_1)t}\frac{\lambda^{m-k_0}}{\Gamma(k^*+(s-1)(k-k')+1)}\frac{\beta^n}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\
&\times\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}t^{m+n-k_0+i}\int_{\tau=0}^{t}(\upsilon t)^{k^*+(s-1)(k-k')+i}(t-\upsilon t)^{m+n-(s-1)(k-k')-k^*-k_0-1}t\ d\upsilon \\
&= e^{-(\lambda+\mu_1)t}\frac{\lambda^{m-k_0}}{\Gamma(k^*+(s-1)(k-k')+1)}\frac{\beta^n}{\Gamma(m+n-(s-1)(k-k')-k^*-k_0)} \\
&\times\sum_{i=0}^{\infty}\frac{\mu_1^{n+i}}{i!}t^{m+n-k_0+i} \\
&\times\frac{\Gamma(k^*+(s-1)(k-k')+i+1)\Gamma(m+n-(s-1)(k-k')-k^*-k_0)}{\Gamma(m+n-k_0+i+1)}
\end{aligned}
$$

B.8 Derivation of Pdf of the Total Vacation Period of IBP of $M/C_2/1$ Under (k', k) Control Policy and Ending in Busy Period

Suppose $\tau = \tau_1 + \tau_2 + \ldots + \tau_s$, where τ the total time in vacation. The probability density function of τ is given by

$$f_{s(k-k')} = \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}\lambda^{k-k'}\tau_y^{k-k'-1}}{\Gamma(k-k')} d\tau_y = \frac{e^{-\lambda\tau}\lambda^{s(k-k')}}{(\Gamma(k-k'))^s} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \tau_y^{k-k'-1} d\tau_y$$

The integration can be done step by step.

First integrate over τ_s, where $\tau_s = \tau - \tau_1 + \tau_2 + \ldots + \tau_{s-1}$.

$$\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}} \tau_s^{k-k'-1} d\tau_s = \frac{\tau_s^{k-k'}}{k-k'}|^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1}} =$$
$$\frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k-k'}}{k-k'}$$

Next for integration over τ_{s-1}, we calculate second integration as follows.

$$\int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}} \frac{(\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-1})^{k-k'}}{k-k'} \tau_{s-1}^{k-k'-1} d\tau_{s-1}$$
$$= \frac{1}{k-k'} \int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'}$$
$$\times \left(1 - \frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k-k'} \tau_{s-1}^{k-k'-1} d\tau_{s-1}$$
$$= \frac{1}{k-k'} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{k-k'} \int_0^{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}}$$
$$\times \left(1 - \frac{\tau_{s-1}}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}\right)^{k^*} \tau_{s-1}^{k-k'-1} d\tau_{s-1}$$

Let $u = \frac{\tau_1}{\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2}}$, $d\tau_{s-1} = (\tau-\tau_1-\tau_2-\ldots-\tau_{s-2}-\tau_{s-2})du$

Then the above right hand side equation becomes

$$\frac{1}{k-k'} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{2(k-k')} \int_0^1 (1-u)^{k-k'} u^{k-k'-1} du$$
$$= \frac{1}{k-k'} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{2(k-k')} \frac{\Gamma(k-k'+1)\Gamma(k-k')}{\Gamma(2(k-k')+1)}$$
$$= \frac{(\Gamma(k-k'))^2}{\Gamma(2(k-k')+1)} (\tau-\tau_1-\tau_2-\ldots-\tau_{s-3}-\tau_{s-2})^{2(k-k')}$$

We need to continue to calculate the integration over $\tau_{s-2}, \tau_{s-3}, \ldots, \tau_2, \tau_1$.

For integration over τ_1, we calculate

$$\int_0^{\tau} \frac{(\Gamma(k-k'))^{s-1}}{\Gamma((s-1)(k-k')+1)} (\tau-\tau_1)^{(s-1)(k-k')} \tau_1^{k-k'-1} d\tau_1$$
$$= \frac{(\Gamma(k-k'))^{s-2}}{\Gamma((s-1)(k-k')+1)} \int_0^{\tau} \tau^{(s-1)(k-k')} \left(1 - \frac{\tau_1}{\tau}\right)^{(s-1)(k-k')} \tau_1^{k-k'-1} d\tau_1$$

Let $u = \frac{\tau_1}{\tau},\ d\tau_1 = \tau du$

Then the above right hand side equation becomes

$$
\begin{aligned}
&= \frac{(\Gamma(k-k'))^{s-1}}{\Gamma((s-1)(k-k')+1)} \int_0^\tau \tau^{(s-1)(k-k')}(1-u)^{(s-1)(k-k')}(u\tau)^{k-k'-1}\tau du \\
&= \frac{(\Gamma(k-k'))^{s-1}}{\Gamma((s-1)(k-k')+1)} \tau^{s(k-k')} \int_0^1 (1-u)^{(s-1)(k-k')} u^{k-k'-1} du \\
&= \frac{(\Gamma(k-k'))^{s-1}}{\Gamma((s-1)(k-k')+1)} \tau^{s(k-k')} \frac{\Gamma((s-1)(k-k')+1)\Gamma(k-k')}{\Gamma(s(k-k')+1)} \\
&= \frac{(\Gamma(k-k'))^{s}}{\Gamma(s(k-k')+1)} \tau^{s(k-k')}
\end{aligned}
$$

Finally we get

$$
f_{s(k-k')} = \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \frac{e^{-\lambda\tau_y}\lambda^{k-k'}\tau_y^{k-k'-1}}{\Gamma(k-k')} d\tau_y = \frac{e^{-\lambda\tau}\lambda^{s(k-k')}}{(\Gamma(k-k'))^s} \int_{\tau_1} \cdots \int_{\tau_s} \prod_{y=1}^{s-1} \tau_y^{k-k'-1} d\tau_y
$$

$$
\frac{e^{-\lambda\tau}\lambda^{s(k-k')}}{(\Gamma(k-k'))^s} \frac{(\Gamma(k-k'))^s}{\Gamma(s(k-k')+1)} \tau^{s(k-k')} = \frac{e^{-\lambda\tau}\lambda^{s(k-k')}}{\Gamma(s(k-k')+1)}
$$

which yields

$$
f_{s(k-k')(\tau)} = \frac{e^{-\lambda\tau}\lambda^{s(k-k')}}{\Gamma(s(k-k')+1)}
$$

B.9 The Derivation of the Integral Part of $f^3_{k_0;(k',k)}(t)$

For the last part of $f^3_{k_0;(k',k)}(t)$ i.e. the double integral, we find

$$\begin{aligned}
&\int_0^t \int_0^{t-\tau} e^{\mu_2\tau} e^{(\mu_2-\mu_1)t_1} t_1^{N_1-1} \tau^{s(k-k')} (t-\tau-t_1)^{N_2-1} dt_1 d\tau \\
&\int_0^t \int_0^{t-\tau} \sum_{i=0}^{\infty} \frac{(\mu_2\tau)^i}{i!} \tau^{s(k-k')} \sum_{j=1}^{\infty} \frac{((\mu_2-\mu_1)t_1)^j}{j!} t_1^{N_1-1} (t-\tau-t_1)^{N_2-1} dt_1 d\tau \\
&\sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \int_0^t \tau^{s(k-k')+i} \\
&\times \left(\int_0^{t-\tau} t_1^{N_1+j-1} (t-\tau)^{N_2-1} \left(1 - \frac{t_1}{t-\tau}\right)^{N_2-1} dt_1 \right) d\tau.
\end{aligned} \tag{B.7}$$

By substituting $u = \frac{t_1}{t-\tau}$ in (B.7), we obtain

$$\begin{aligned}
&\int_0^t \int_0^{t-\tau} e^{\mu_2\tau} e^{(\mu_2-\mu_1)t_1} t_1^{N_1-1} \tau^{s(k-k')} (t-\tau-t_1)^{N_2-1} d\tau dt_1 \\
&= \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \int_0^t \tau^{s(k-k')+i} (t-\tau)^{N_1+N_2+j-1} \\
&\times \left(\int_0^1 u^{N_1+j-1} (1-u)^{N_2-1} du \right) d\tau \\
&= \sum_{j=1}^{\infty} \frac{\mu_2^j}{j!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^i}{i!} \int_0^t \tau^{s(k-k')+i} (t-\tau)^{N_1+N_2+j-1} \beta(N_1+j, N_2) d\tau.
\end{aligned} \tag{B.8}$$

Suppose $\upsilon = \frac{\tau}{t}$, then (B.8) can be written as

$$\begin{aligned}
&\sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} \int_0^1 (\upsilon t)^{s(k-k')+i} (t-\upsilon t)^{N_1+N_2+j-1} \beta(N_1+j, N_2) t d\upsilon \\
&= \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j=1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} t^{N_1+N_2+s(k-k')+i+j} \beta(N_1+j, N_2) \int_0^1 \upsilon^{s(k-k')+i} \\
&\times (1-\upsilon)^{N_1+N_2+j-1} d\upsilon \\
&= \sum_{i=1}^{\infty} \frac{\mu_2^i}{i!} \sum_{j1}^{\infty} \frac{(\mu_2-\mu_1)^j}{j!} t^{N_1+N_2+s(k-k')+i+j} \beta(N_1+j, N_2) \\
&\times \beta(s(k-k')+i+1, N_1+N_2+j)
\end{aligned} \tag{B.9}$$

References

[1] Agarwal, M., K. Sen, and B. Borkakaty. Lattice path approach for busy period density of $GI_b/G/1$ queues using C_2 Coxian distributions. *Statistical Theory and Practice*, 1(2), 2007.

[2] Agarwal, M., K. Sen, and B. Borkakaty. Lattice path approach for busy period density of M/G/1 queues using C_3 Coxian distribution. *Applied Mathematical Modelling*, 31(10):2062–2079, 2007.

[3] Agarwal, R. P. and J. H. Dshalalow. New fluctuation analysis of D-policy bulk queues with multiple vacations. *Mathematical and Computer Modelling*, 41:253–269, 2005.

[4] Ausin, M. C., M. P. Wiper and R. E. Lillo. Bayesian estimation for the M/G/1 queue using a phase-type approximation. *Journal of Statistical Planning and Inference*, 1-2(118):83–101, 2004.

[5] Banderier, C., P. Flajolet. Basic analytic combinatorics of directed lattice paths. *Theoretical Computer Science*, 281:37–80, 2002.

[6] Böhm, W. Lattice path counting and the theory of queues. Technical Report 74, Vienna University of Economics and Business, Vienna University of Economics and Business, Vienna, Department of Statistics and Mathematics, 74, WU, 2008.

[7] Böhm, W. and S. G. Mohanty. The transient solution of M/M/1 queues under (M,N)-policy: A combinatorial approach. *Journal of Statistical Planning and Inference*, 34(1):23–33, 1993.

[8] Bell, C. E. Characterization and computation of optimal policies for operating an M/G/1 queueing system with removable server. *Operations Research*, 21:1281–1289, 1971.

[9] Bingham, N. H. and S. M. Pitts. Nonparametric inference from M/G/l busy periods. *Stochastic Models*, 2(15):247–272, 1999.

[10] Borkakaty, B., K. Sen, and M. Agarwal. Busy period analysis of C_2/C_{2b} /1 queue: Lattice path approach. *Mathematical and Computer Modelling*, 50(7):1067–1082, 2009a.

[11] Borkakaty, B., M. Agarwal and K. Sen. Lattice path approach for busy period density of G/G/1 queues using C_2 Coxian distributions. *Applied Mathematical Modelling*, pages 11–111, 2009b.

[12] Bruneel, H. Performance of discrete time queueing systems. *Computers Operations Research*, 20:303–320, 1993.

[13] Bruneel, H. and B. G. Kim. *Discrete Time Models for Communication Systems Including ATM.* Kluwer Academic Plublishers, Boston, 1993.

[14] Bruneel, H. and I. Wuyts. Analysis of discrete-time multiserver queueing models with constatnts service times. *Operations Research Letters*, 15:231–236, 1994.

[15] Buzacott, J. A. and J. G. Shanthikumar. *Stochastic Models of Manufacturing Systems.* Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1993.

[16] Buzeteanu, S. N. and V. Domocos. Polynomial identities from weighted lattice path counting. *Discrete Mathematics*, 150(1-3):421–425, 1996.

[17] Chae, K. C. and D. E. Lim. Busy period analysis for the N-policy GI/M/c queue. *Journal of the Korean Statistical Society*, 37(3):285–290, 2008.

[18] Chae, K. C. and S. J. Kim. Busy period analysis for the G/M/1 queue with exponential vacations. *Operations Research Letters*, 35(1):114–118., 2007.

[19] Chao, X. and A. Rahman. Analysis and computational algorithm for queues with state-dependent vacations II: $M_n/G/1/$. *Jrl Syst Sci & Complexity*, 19:191–210, 2006.

[20] Choudhury, G. Analysis of the $M_x/G/1$ queueing system with vacation times. *Sankhya : The Indian Journal of Statistics*, 64(B):37–49, 2002.

[21] Choudhury, G. and L. Tadj. An M/G/1 queue with two phases of service subject to the server breakdown and delayed repair. *Applied Mathematical Modelling*, 33:2699–2709, 2009.

[22] Chu, Y. K. and J. C. Ke. Confidence intervals of mean response time for an M/G/1 queueing system: Bootstrap simulation. *Applied Mathematics and Computation*, 1(180):255–263, 2006.

[23] Cohen, J. W. *The Single Server Queue*. North-Holland, Amsterdam,, 1969.

[24] Cox, D. R. A use of complex probabilities in the theory of stochastic processes. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 51, pages 313–319, 1955.

[25] Cox, D. R. and H. D. Miller. *An Introduction to the Theory of Stochastic Processes*. Methuen & Co, London, 1965.

[26] Doshi, B. T. Queueing systems with vacations - a survey. *Queueing Systems*, 1:29–66, 1986.

[27] Doshi, B. T and H. Heffes. Overload performance of several processor queueing disciplines for M/M/1 queue. *IEEE Transactions on Communications*, 34(6):538–546, 1986.

[28] Ericksen, L. Lattice path combinatorics for multiple product identities. *Journal of Statistical Planning and Inference*, 140:2213–2226, 2010.

[29] Fallon, J., S. Gao and H. Niederhausen. Proof of a lattice paths conjecture connected to the tennis ball problem. *Journal of Statistical Planning and Inference*, 140:2227–2229, 2010.

[30] Fayolle, G., R. Iasnogorodski, V. Malyshev. *Random Walks in the Quarter-plane.* Springer, Berlin, 1999.

[31] Fearnhead, P. Filtering recursions for calculating likelihoods for queues based on inter-departure time data. *Statistics and Computing*, 3(14):261–266, 2004.

[32] Feller, W. *An Introduction to Probability Thery and its Applications*, volume 1. John Wiley, New York, 3rd edition, 1968.

[33] Fiems, D. and H. Bruneel. Discrete-time queueing systems with Markovian preemptive vacations. *Mathematical and Computer Modelling*, 2012. Article in Press.

[34] Fiems, D., S. DeVuyst and H. Bruneel. The combined gated-exhaustive vacation system in discrete time. *Performance Evaluation*, 49(1-4):227–239, 2002.

[35] Garg, R. Busy period analysis of a two-unit warm standby system with imperfect switch. *Microelectronics and Reliability*, 25(4):675–679, 1985.

[36] Grassman, W. K. Transient solutions in Markovian queueing systems. *Computers and Operations Research*, 4(1):47–56, 1977.

[37] Gross, D., and C. M. Harris. *Fundamentals of Queueing Theory: Third edition.* John Wiley & Sons, Canada, 1998.

[38] Gupta, U. C. and T. S. S. Srinivisa Rao. A recursive method to compute the steady state probabil- ities of the machine interference model: M/G/1/K. *Computers and Operations Research*, 21:597–605, 1994.

[39] Gupta, U. C. and T. S. S. Srinivisa Rao. On the M/G/1 machine interference model with spares. *European Journal of Operations Research*, 89:164–171, 1996.

[40] Hansen, M. B. and S. M. Pitts. Nonparametric inference from the M/G/1 workload. *Bernoulli*, 4(12):737–759, 2006.

[41] Harris, C. M., W. G. Marchal and R. F. Botta. A note on generalized hyper-exponential distributions communications in statistics. *Stochastic Models*, 8(1):179–191, 1992.

[42] Hernandez-Lerma, O. and S. I. Marcus. Adaptive control of service in queueing systems. *Systems & Control Letters*, 3(5):283–289, 1983.

[43] Hernandez-Lerma, O. and S. I. Marcus. Optimal adaptive control of priority assignment in queueing systems. *Systems & Control Letters*, 4(2):65–72, 1984.

[44] Heyman, D. P. Optimal operating policies for M/G/1 queueing system. *Operations Research*, 16:362–382, 1968.

[45] Insua, D. R., M. Wiper and F. Ruggeri. Bayesian analysis of M/E_r/1 and M/H_k/1 queues. *Queueing Systems*, 30(33):289–308, 1998.

[46] Iversen, D. R. and B. F. Nielsen. Some properties of Coxian distributions with application. *Modelling Techniques and Tools for Performance Analysis*, 85(61-66), 1985. N. Abu El Ata (Editor).

[47] Jain, S. Estimating changes in traffic intensity for M/M/1 queueing systems. *Micro-electronics and Reliability*, 11(35):1395–1400, 1995.

[48] Kao, E. P. C. and K. S. Narayanan. Analyses of an M/M/n queue with server's vacations. *European Journal of Operational Research*, 54(2):256–266, 1991.

[49] Ke, J. -C., L. Chuen-Horng, Y. Jyun-Yi and Z. Zhe George. Optimal (d, c) vacation policy for a finite buffer M/M/c queue with unreliable servers and repairs. *Applied Mathematical Modelling*, 33:3949–3962, 2009.

[50] Kekre, H. B., C. L. Saxena and M. Khalid . Buffer behaviour for mixed arrivals and single server with random interruptions. *IEEE Transactions on Connunications*, 28(1):59–64, 1980.

[51] Khosgoftaar, T. M. and H. G. Perros. *A Comparison of Three Methods of Estimation for Approximating General Distributions by a Coxian Distribution: Modelling techniques and performance evaluation in: S. Fdida, G.*

Pujolle (Eds.), volume 1. Elsevier Publishers B.V., North-Holland, Amsterdam, 1987.

[52] Kleinrock, L. *Queueing Systems: Volume 1, Theory.* John Wiley & Sons, 1975, New York.

[53] Kleinrock, L. *Queueing Systems, Volume 2: Computer applications.* John Wiley & Sons, 1976, New York.

[54] Krinik, A. Transient probabilities of a single server priority queueing system. *Journal of Statistical Planning and Inference*, 101(1-2):185–190, 2002.

[55] Kumar, B. K. and S. P. Madheswari. An M/M/2 queueing system with heterogeneous servers and multiple vacations. *Mathematical and Computer Modelling*, 41(13):1415–1429, 2005.

[56] Larson, R. C. and A. R. Odoni. *Urban Operations Research.* Prentice-Hall, New Jersey, 1981.

[57] Lee, J. E. and Y. Hong. A stock rationing policy in a (s,S) controlled stochastic production system with 2-phase Coxian processing times and lost sales. *Int. J. Production Economics*, 83:299–307, 2003.

[58] Levy, Y. and U. Yechiali. An M/M/s queue with servers' vacations. *INFOR*, 14:153–163, 1976.

[59] Li, W., D. Shi and X. Chao. Reliability analysis of M/G/1 queueing systems with server breakdowns and vacations. *J. Appl. Prob.*, 34:546–555, 1997.

[60] Liu, M., Y. Ma and B. Deng. The system size distribution for M/G/1 queueing system under N-policy with startup/closedown. *iBusiness*, 2(4):363–369, 2010.

[61] Machihara, F. Carried traffic estimate errors for delay systems. *Electronics and Communications in Japan*, 67(12):49–58, 1984.

[62] Marshall, A. H. and M. Zenga. Recent developments in fitting Coxian phase-type distributions in healthcare. *Applied Stochastic Models and Data Analysis*, 1:482–485, 2009.

[63] McGrory, C. A., A. N. Pettitt, M. J. Faddy. A fully Bayesian approach to inference for coxianphase-type distributions with covariate dependent mean. *Computational Statistics & Data Analysis*, 53(12):4311–4321, October 2009.

[64] Medhi, J. *Stochastic Models in Queueing Theory.* Academic Press, New York, 1991.

[65] Mendelson, H. Lattice path combinatorics and linear probing. *Journal of Statistical Planning and Inference*, 14(1):79–93, 1986.

[66] Mercankosk, G., G. M. Nair and W. J. Soet. Combinatorial techniques for $m/g/1$ type queues. *J. Appl. Prob.*, 38:722–736, 2001.

[67] Mohammadi, A. and M. R. Salehi-Rad. Bayesian inference and prediction in an M/G/1 with optional second service. *Communications in Statistics Simulation and Computation*, 3(41):419–435, 2012.

[68] Mohanty, S. G. *Lattice Paths Counting and Applications.* Academic Press, New York, 1979.

[69] Mohanty, S. G. and W. Panny. A discrete time analog of M/M/l queue and the transient solution: A geometric approach. *Sankhya, Ser. A.*, 52:364–370, 1990.

[70] Muto, K., H. Miyazaki, Y. Seki, Y. Kimura and Y. Shibata. Lattice path counting and M/M/c queueing systems. *Queueing Systems*, 19:193–214, 1995.

[71] Narahari, Y. and N. Viswanadham. Transient analysis of manufacturing systems performance. *IEEE Transaction on Robotics and Automation*, 10(2):230–244, 1994.

[72] Neuts, M. F. *Matrix-Geometric Solutions in Stochastic Models: An algorithmic approach.* The Johns Hopkins University Press, Baltimore and London, 1981.

[73] Neuts, M. F. *Structured Stochastic Matrices of M/G/1: Type and their applications*. Marcel Dekker, Inc., New York, 1989.

[74] Panwar, S. S., D. Towsley and J. K. Wolf. Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service. *Journal of the* ACM *(*JACM*)*, 35(4):832–844, October 1988.

[75] Parthasarathy, P. R. A transient solution to an M/M/1 queue: a simple approach,. *Adv. Appl. Prob.*, pages 997–998, 1987.

[76] Parthasarathy, P. R. and M. Sharafali. Transient solution to the many server poisson queue: a simple approach. *J. Appl. Prob.*, 26:584–594, 1989.

[77] Perry, D., W. Stadje and S. Zacks. Busy period analysis for M/G/1 and G/M/1 type queues with restricted accessibility. *Operations Research Letters*, 27:163–174, 2000.

[78] Rao, S. S. Queueng with balking and reneging in M/G/1 systems. *Metrika*, 12:173–188, 1968.

[79] Rodrigo, A. Estimators of the retrial rate in M/G/1 retrial queues. *Asia Pacific Journal of Operational Research*, 23(2):193–201, 2006.

[80] Rodrigo, A. and M. V. G. Falin. A new description of the M/G/1 retrial queue. *European Journal of Operational Research*, 104:231–240, 1998.

[81] Saaty, T. L. Time dependent solution of many server poisson queue. *Oper. Res.*, 8:755–772, 1960.

[82] Sen, K. Lattice path approach to transient analysis of M/G/1/N non-Markovian queues using Cox distributions. *Journal of Statistical Planning and Inference*, 101:133–147, 2002.

[83] Sen, K. and J. L. Jain. Combinatorial approach to Markovian queueing models. *Journal of Statistical Planning and Inference*, 34(2):269–279, 1993.

[84] Sen, K. and M. Agarwal. Transient busy analysis of initially non-empty M/G/1 queues-lattice path approach. *Advances in Combinatorial Meth-*

ods and Applications to Probability and Statistics, 34:301–315, 1997. Balakrisnhnan, N. (ed.).

[85] Sen, K. and M. Agarwal. Lattice paths combinatorics applied to transient queue length distribution of C_2/M/1 queues and busy period analysis of bulk queues C_b/M/1. *Journal of Statistical Planning and Inference*, 100(2):365–397, 2002.

[86] Sen, K. and R. Gupta. M/M/1 T-policy queues with service control: a study of transient behaviour. In *Proc. 1st Int. Calcutta symp. on Probability and Statistics*, pages 20–32, India, December 1993.

[87] Sen, K. and R. Gupta. Transient analysis of threshold T-policy M/M/1 queues with server control. *Sankhya series B*, 56:39–51, 1994.

[88] Sen, K. and R. Gupta. Transient solution of M_b/M/1 system under threshold control policies. *J. Statist. Res.*, 30:109–120, 1996.

[89] Sen, K., J. L. Jain and J. M. Gupta. Lattice path approach to transient solution of M/M/1 with $(0, k)$ control policy. *Journal of Statistical Planning and Inference*, 34:259–267, 1993.

[90] Sridharan, V. and P. Mohanavadivu. Stochastic behaviour of a two-unit standby system with two types of repairmen and patience time. *Mathematical and Computer Modelling*, 28(9):63–71, 1998.

[91] Tadj, L. On an M/G/1 quorum queueing system under T-policy. *Operational Research Society*, (54):466–471, 2003.

[92] Takacs, L. Delay distribution for one line with Poisson input, general holding times and various orders of service. *Bell System Technical Journal*, 42:487–504, 1963.

[93] Takacs, L. *Combinatorial Methods in the Theory of Stochastic Processes.* John Wiley and Sons, New York, 1967.

[94] Takagi, H. *Queueing Analysis : A foundation of performance evaluation, volume 1: vacation and priority systems.* John Wiley & Sons, New York, 1991.

[95] Takagi, H. *Queueing Analysis : A foundation of performance evaluation, volume 2: finite systems.* John Wiley & Sons, New York, 1993.

[96] Takagi, H. *Queueing Analysis; A foundation of performance evaluation, volume 3: discrete-time systems.* Elsevier Science Publishers, Amsterdam, 1993.

[97] Team, R. D. C. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2009.

[98] Thiagarajan, T. R. and C. M. Harris. Statistical tests for exponential service from M/G/1 waiting-time data. *Naval Research Logistics Quarterly*, 26(3):511–520, 1979.

[99] Tijm, H. C. *Stochastic Modeling and Analysis.* Wiley, New York, 1986.

[100] Towsley, D. The analysis of a statistical multiplexer with nonindependent arrivals and errors. *IEEE Transactions on Communications*, 28(1):65–72, 1980.

[101] Vidalis, M. I. and H. T. Papadopoulos. Markovian analysis of production lines with Coxian-2 service times. *International Transactions in Operational Research*, 6(5):495–524, 1999.

[102] Wang, K. H. and H. M. Huang. Optimal control of an M/E/1 queueing system with a removable service station. *Journal of Operational Research Society*, 46:1014–1022, 1995.

[103] Wang, K. H. and J. C. Ke. A recursive method to the optimal control of an M/G/1 queueing system with finite capacity and infinite capacity. *Applied Mathematical Modelling*, 24:899–914, 2000.

[104] Wazalwar, N. M., M. V. Khaparde. On transient behaviour of $m/c_2/1$ independent queueing model with $(0, k)$ control policy using lattice path technique. *Bulletin of Pure & Applied Sciences, Mathematics and Statistics*, 30e(2):231–250, 2011.

[105] While, J. A., J. W. Schmidt and G. K. Bennett. *Analysis of Queueing System*. Academic Press inc, New York, 1975.

[106] Wittevrongel, S. and H. Bruneel. Discrete-time queues with correlated arrivals and constant service times. *Computers and Operations Research*, 26:93–108, 1999.

[107] Woodward, M. E. *Communication and Computer Networks*. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[108] Yadin, M. and P. Naor. Queueing system with removable service station. *Operational Research Quarterly*, 14:393–405, 1963.

[109] Zhang, Q. and X. Xu. Confidence intervals of performance measures for an M/G/1 queueing system. *Communications in Statistics Simulation and Computation R*, 3(39):501–516, 2010.