October 2018

Statistical Monitoring of Queuing Networks

Yaren Bilge Kaya
University of South Florida, yarenbilgekaya@gmail.com

Statistical Monitoring of Queuing Networks

by

Yaren Bilge Kaya

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Industrial Engineering
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Date of Approval:
October 26, 2018

Keywords: Jackson's networks, Queuing analysis, Markov Chain
Statistical Process Control, CUSUM Charts

DEDICATION

This study is dedicated to my little brother, Engin Can Kaya...

ACKNOWLEDGEMENTS

I would like to thank my project adviser Assistant Professor Devashish Das for his guidance, analytical and constructive critiques, open door policy and providing endless encouragement. This accomplishment would not be possible without his support.

Besides my advisor, I am also grateful to my thesis committee: Dr. Susana Lai-Yuen, Dr. Mingyang Li, Dr. Micheal Weng for their insightful comments and valuable guidance. In addition to my Professors, I would like to thank Yinqang Kuang for her help.

Last but not least, I would like to thank my parents and my brother for providing me unfailing support throughout my years of study and through the process of researching and writing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Queuing systems are important parts of our daily lives and to keep their operations at an efficient level they need to be monitored by using queuing Performance Metrics, such as average queue lengths and average waiting times. On the other hand queue lengths and waiting times are generally random variables and their distributions depend on different properties like arrival rates, service times, number of servers [23]. We focused on detecting the change in service rates in this report. Therefore, we monitored queues by using Cumulative Sum(CUSUM) charts based on likelihood ratios and compared the Average Run Length values of different service rates.

CHAPTER 1: INTRODUCTION

In stochastically dynamic service environments, modeling and optimizing the environment and parameters is in the scope of research. These environments can be represented as queuing networks. According to Thomopoulos(2012), queuing systems are generally combinations of input population, arrivals, queue (a line that entities wait to get their service), service facilities (servers) and departures [2]. Applications of queuing networks vary and it's possible to see their applications in different research fields. From previous research, we saw that there are studies on the application of queuing theory in logistics [12], manufacturing [17], healthcare systems [18], the supply of materials [28] and so on. Thus, there are different kinds of queuing systems in our everyday lives; in supermarkets, in hospitals or in call centers. Therefore, increasing the service quality of a queuing system will have a tremendous impact on our lives. Service quality can be increased by the following approaches; decreasing wait time or queue length. If we are working on a manufacturing line, increasing the service quality may return us as increasing the productivity, which generally means results in profits. If we are working on healthcare systems, increasing the quality may result decreasing in disease outbreak and may even decrease fatalities in emergency rooms. Therefore, it is necessary to carefully monitor the queuing performance metrics like queue lengths and waiting times [23]. For instance, in an emergency room, if we face an abnormal waiting line we may need to increase the number of service stations or the number of emergency nurses. It is shown in Johnson's work [21] that emergency room crowding has negative effects on practices of emergency nurses and also on patients itself. Therefore, monitoring the processes and spotting changes on time can be vital sometimes.

There are different kinds of queuing networks both in theory and in the application: series(tandem) networks, Jackson networks, BCMP networks are some examples. All of these examples of networks are combinations of queues with different properties and these individual queues within networks as "nodes" of the network. In general, properties of a queue are: arrival rate distribution, service rate distribution, number of servers in the queue, population of the inflow and the queuing policy (First Come First Served, Last Come First Served, ...). Since queuing networks behave as a union of individual queues, they carry the characteristics of the queues within them. In addition to these characteristics, there are also routing probabilities which defines probability of going from one queue to another whenever a service is finished in the current queue.

In tandem networks there are multiple queues connected to each other in series. If we have K nodes in a series network, inflow of the i^{th} node is equivalent to outflow of the $(i-1)^{\text{th}}$ node. Therefore we can say that in series networks the routing probability of going from node $i-1$ to i is equal to 1, where $i = \{1, 2, 3, \ldots, K\}$. However, Jackson networks generally consist of multiple queues connected to each other with some routing probabilities with $(0, 1]$. The special property of Jackson is explained in Gelenbe(1987), all the queues in these networks behave as independent and individual queues [3]. Thus, we can say that the service rate of the first queue of the network is assumed to be independent of the service rate of the second queue of the network. This property of Jackson networks helps us find the steady state properties easier. There are two types of Jackson networks in literature they are called as: open and closed Jackson networks [1]. In the closed network, there is no external arrival to any of the queues and there is no departure from any of the queues. However in the open Jackson networks, external arrival can happen to any of the queues and entities can leave the system from any of the queues. Main networks that we have worked on in this report are Jackson Networks and we are going to focus on queues with exponentially distributed service rates, one server at a queue and the arrival rates are going to follow Poisson distribution; in Kendall's notation [8] they are called as M/M/1 queues.

Queuing networks can be examined as Continuous Time Markov Chains (CTMC). Because as the Markov Chain definition: probability of the next event is only depends on the previous event, not the events before that [22]. And generally in queuing theory states are defined as the number of entities in a queue where the state space is defined as: [0, maximum number of entities in a queue]. However, it's difficult to observe the queue lengths continuously; therefore for changing the CTMC to Discrete Time Markov Chain (DTMC) embedded property can be applied. In other words, looking at time points where a transition occurs instead of the time itself allows us to define CTMC queuing network as a DTMC model. While implementing this methodology main assumption is: there cannot be 2 transitions at the same time in a queuing network. There can either be a "birth" process, "death" process or a switch between nodes. Birth is defined as having an external arrival to the system and the death is defined as having a departure from the system. Switch between nodes can happen only if the queues have a routing probability different than zero from one to another. In other words: a death in i^{th} node and a birth in $(i+1)^{\text{th}}$ node at the same time. In our report we used this method to obtain the queue lengths of each queue in the network and this also helped us while finding the transition probabilities between states.

Improving service quality in queuing systems is highly difficult since the nature of these systems is stochastic and hard to monitor. In addition, most of the time observations from queuing networks are not complete. For example in an emergency room, if we define emergency nurses and doctors as servers; we may find the number of people waiting to be served from a specific server but it will be hard to detect the exact time that they have spent in the queue or in the system. Because of these problems, not so many research conducted on statistical process control of queuing networks. Most of the studies are on examining the changes in queuing performance metrics like average number of entities in a queue or average waiting time in a queue lead them to spot abnormalities in the operations. Bhat, Rao(1972) worked on controlling traffic intensity in queuing systems by working on M/G/1 and GI/M/1 queues. In their study, they found upper and lower control limits to observe the out of control points [6] by using statistical quality control techniques. Whenever a point is under lower control point or above upper control point they accepted that the system has gone out of control. Long after them Shore(2006) used attributes control charts to monitor the queue lengths in G/G/S queues [7]. While using attributes charts he also handled the skewness in queue lengths and obtained successful results. Chen, Yuan, Zhou(2011) extended Bhat, Rao(1972)'s work by comparing average run length (ARL) values of two different charts: nL charts and warning zone charts. Average run length can be defined as the average of expected number of samples taken before an Out of Control(OC) point observed. Later on Chen, Zhou(2014) used Cumulative Sum(CUSUM) Charts based on likelihood ratios to monitor the performance of mainly M/M/1 queues they also extended their research by applying the methods to M/G/1 and G/G/1 queues. They mentioned in their research that, as Lucas (1985) [31], and Ryu, Wan and Kim (2010) [29] CUSUM charts are more sensitive and they detect changes faster than Shewhart Charts. With the help of these studies we used CUSUM Charts based on likelihood ratios with different designs. To obtain a solution we compared the average run lengths of these charts in this report. We are conducting our research especially on detecting changes in service rates, therefore in simulation, we only changed the service rates of our design parameters.

In this work we make the following contributions: (i) We detect changes in service rates faster by using CUSUM charts based on likelihood ratios (ii) We monitored queuing Performance Metrics on queuing networks instead of individual queues different from previous studies (iii) We used average run length while comparing the performance of CUSUM charts. Thus, with these contributions, our aim is to follow and expand previous research.

The remaining of this report organized as follows. Chapter 2, explains basic queuing networks, their properties and then focuses on steady state and transition probabilities of Open Jackson networks which are used to find likelihood ratios. Chapter 3 gives a brief introduction to likelihood estimations and explains likelihood ratios while computing CUSUM Charts. Chapter 4 discusses the simulation results and compares ARL performances of different design parameters. Finally, the report concludes with the explanation of future works in Chapter 5.

CHAPTER 2: QUEUING NETWORKS

You may find an introduction to some basic queuing networks in this chapter. Manufacturing facilities, assembly lines, communication networks, and health care systems are everyday life examples of these queuing networks.

2.1 Basic Explanation of Queuing Networks

These networks consist of one or more nodes, where each node represents a service. This service can be served by one or more servers at each node. In most of the cases, customers/entities enter the system from a node, travel some other nodes and then depart from the system. But this may change; customers may skip some nodes, they may visit a node more than one time or they may stay in the system forever.

We generally name the queues by using the following Kendall notation [8] A/B/C/D/E

• A stands for arrival time distribution

• B stands for service time distribution and if we have more than one server at a node it also shows whether the servers are identical or not.

• C stands for the number of servers at a node

• D stands for the maximum number of units in network (finite or infinite)

• E stands for the queuing policy (FIFO, LCLS, shortest process time)

In this report we will mainly focus on open Jackson networks which have:

• Poisson process in their arrivals.

• Exponentially and independently distributed service times at nodes.

• Routing probabilities.

• Infinite queue capacity.

• Utilization ≤ 1 for all servers.

We're going to start explaining queuing networks by first explaining series (tandem) queues. In these networks, customers enter the system only from the first node and departs from the last node by following the series of nodes. Also, the flow of the system is always directed in one direction. You may see the basic representation in Figure 2.1.

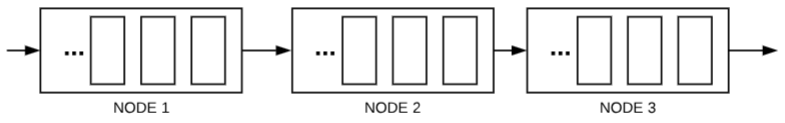

Figure 2.1: Basic example of a series network

Other than series queues, there are more complex queuing networks like Jackson networks. There are two types of Jackson networks: one of them is called open Jackson networks and the other one is closed Jackson networks. The closed networks have a fixed number of population which we can call as N. The customers or items circulate continuously in the system and they never leave. In the open networks, customers or items come from external sources and they eventually depart from the system. Figure 2.2 is a basic example of an open Jackson networks and Figure 2.3 is an example of a closed network.

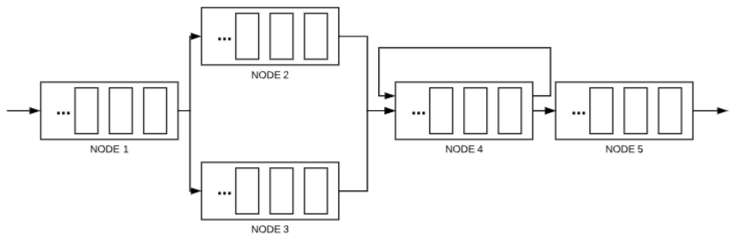

Figure 2.2: Basic example of an open Jackson network

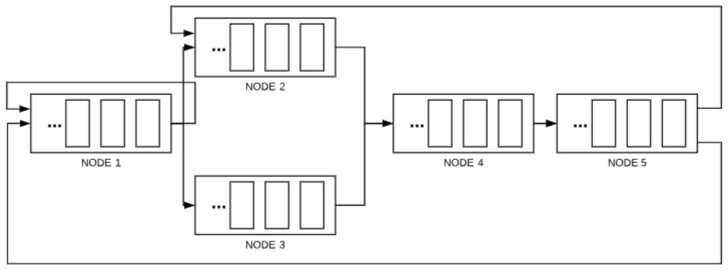

Figure 2.3: Basic example of a closed Jackson network

2.2 Series Networks

In these networks, arrival process of the queues is generally Poisson with mean λ and service times are exponentially distributed with the mean of $1/\mu_i$ where $i = 1, 2, 3...K$ and i stands for the number of nodes in the network. If there is more than one server in the nodes we assume that their service times are independent and identical. There is no capacity in the queues. An everyday example of this concept is assembly lines. All products in the assembly lines enter the system from the beginning node and depart from the last node.

While analysing these kinds of queues we can behave each node as a separate queue. Because we assume that output distribution of the previous node is equal to the input distribution of the next node. For example, if we have c_1 servers at 1^{st} node and c_2 servers at 2^{nd} node; we can say that node 1 is a M/M/c_1/∞ model and node 2 is a M/M/c_2/∞ model. Departure time distribution of an M/M/c_i queue is identical to inter-arrival time distribution. Therefore while analysing these kinds of networks we are going to assume that we have separate queues and then we will aggregate them.

One of the most important things while analysing these networks is the traffic equations and we can obtain them by using ”*input* = *output*” relationship.

Let's assume that we have a basic "2 nodes with one server in each queue" network. Flow from node 1 to 2 is exponentially distributed with μ_1 and flow from node 2 to departure is μ_2. Only entering point to the system is from node 1 and only departure point is node 2. Arrivals are Poisson process with the mean λ.

Let N_i be the random variable, which represents the number of customers in the node i at steady state. And we are looking for the joint probability:

$$Pr\{N_1 = n_1, N_2 = n_2, ..., N_k = n_k\} = p_{(n_1,n_2,...,n_k)}$$

$$p_{(n_1,n_2)} = (1-\rho_1)\rho_1^{n_1}(1-\rho_2)\rho_2^{n_2} \quad \text{where} \quad \rho = \lambda/\mu \leq 1 \quad \text{and } n \geq 0 \tag{1}$$

When there is k nodes:

$$p_{(n_1,n_2,...,n_k)} = \prod\nolimits_{i=1}^{k}(1-\rho_i)\rho_i^{n_i} \quad \text{where} \quad \rho_i = \lambda_i/\mu_i \leq 1 \quad \text{and } n_i \geq 0 \ \forall i \tag{2}$$

2.2.1 Average Measures

In order to find the expected waiting time in the queues and the number of customers in the queues or in the system, we need the average measurements you may see below. L stands for the expected number of customers in the system; and L_q stands for the expected number of customers in the queue.

$$L_i = \sum_{i=1}^{k} \frac{\lambda_i}{\mu_i - \lambda_i} \tag{3}$$

By using Little's Law [9] we can also find the expected total time spent on the system which is represented as W.

$$W_i = \frac{L_i}{\lambda_i} \tag{4}$$

2.3 Open Jackson Networks

In this model, customers may enter the system from any node with Poisson distribution with the mean λ. There can be one or more servers at each node with service time exponentially distributed with mean μ_i. If there are more than one servers at each node, we are going to assume that all servers are independent of each other and their service rates are identical. When customers complete the service at node i, there is a probability of r_{ij} that the customer may go to the next (j^{th}) node. Also, customers may leave the system from any node with the probability r_{i0}. In addition to these, we accept that there is no capacity on queues.

Jackson's theorem says that: "In steady state, each node behaves independently as M/M/1 or M/M/c queue with infinite capacity. And the number of customers in different nodes are independent. In addition to that, queues behave like their arrival stream is Poisson." [14] [1]. Therefore, while analysing this model at first we are going to assume that all nodes are separate and then we are going to aggregate our solution.

If we call γ_i as external arrival rate and λ as total arrival rate to that node, we are going to obtain the equation (5) from traffic equations.

$$\lambda_i = \gamma_i + \sum_{i=1}^{K} \lambda_j r_{ij} \text{ where } 1 \leq i \leq K \tag{5}$$

We are going to focus on Open Jackson networks with M/M/1 queues in this report, that's why we are going to give more details about their steady state and transition probabilities below in addition to their average measures.

2.3.1 Average Measures

In this part, you may find the measurements and the formulas for one server model of open Jackson networks. L represents the number of customers in the system and W represents the waiting times in the system. You may convert this model into a M/M/c model by using the necessary formulas.

$$L_i = \frac{\rho_i}{1-\rho_i} \text{ where } \rho_i = \frac{\lambda_i}{\mu_i} \tag{6}$$

$$W_i = \frac{L_i}{\gamma_i} \text{ and therefore, } W_{q_i} = W_i - \frac{1}{\mu_i} \tag{7}$$

In order to find the total waiting time in the system we can sum all the λ_is that we have found since they are independent of each other and we can call it as λ. Then we can use the Equation (8) below.

$$W = \frac{1}{\lambda}\sum_{i=1}^{k} L_i \tag{8}$$

2.3.2 Steady State Probabilities of Open Jackson Networks

In order to find the steady state probabilities of queuing networks we need to use Jackson's theorem. In Jackson's theorem, every node in a network considered as an individual, independent M/M/1 or M/M/c queue therefore, we assume that number of customers in one node is independent of the number of customers in another node. Thus while finding the steady state probabilities, we can use the Equation (9) [4].

$$p_{(n_1,n_2,\ldots,n_k)} = \prod\nolimits_{i=1}^{k}(1-\rho_i)\rho_i^{n_i} \text{ where } \rho_i = \lambda_i/\mu_i \leq 1 \quad \text{and } n_i \geq 0 \ \forall i \tag{9}$$

As you can see from the Equation (9) steady state probabilities of Jackson networks follow geometric distribution with the success probability $(1-\rho_i)$.

2.3.3 Transition Probabilities of Open Jackson Networks

The probability of the occurrence of a transition between two states are called as transition probabilities in Markov Chains. If we would like to examine our queuing networks as Markov chains first we need states and generally, they are defined as the number of people in the queue in queuing theory. These states change depending on time, that's why we can call this process as a continuous time Markov Chain(CTMC). Unfortunately, it is difficult to find transition probabilities by using CTMC, therefore we can transform this system from continuous time to discrete time by just looking at the times where transitions occur. We call these types of Markov Chains as embedded Markov Chains [5] and we can call the points where transitions occur as τ_l where $l = 1, 2, ...L$.

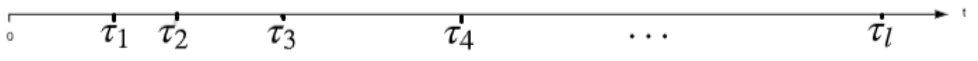

Figure 2.4: Timeline and the points where transitions occur (τ)

If we call $X(t)$ as a continuous time Markov Chain which is a function of time and define functions Y_1 and Y_2 as $Y_1 = X(\tau_1)$ and $Y_2 = X(\tau_2)$ respectively; we can call these Y_1 and Y_2 as discrete time Markov Chains.

First, we are only going to talk about a M/M/1 queue instead of a queuing network to explain the basic logic. We suppose that the process is in state n right now and till a transition occurs it will stay at state n. If we say that the arrivals occur with Poisson distribution with rate λ our average time until an arrival occurs is going to be $1/\lambda$. If the service rate is exponential with the rate μ we can say that average time until a service completion (departure) is going to be exponential with $1/\mu$. We know that the system will leave the state n whenever an arrival or a departure occurs so that the time spent in state n is going to be the minimum of these two events. When we do the calculations we see that the time that system spent in state n will be exponential with rate $1/(\lambda + \mu)$. Thus, the probability of having an arrival is going to be $\lambda/(\lambda + \mu)$ and the probability of having a departure is going to be $\mu/(\lambda+\mu)$. By looking at these values, we can see our system as a discrete time Markov Chain with transition probability of going one state up as $\lambda/(\lambda+\mu)$ and going one state down as $\mu/(\lambda+\mu)$ [1]. We should also remember the important point which is: this method only works if the $\rho < 1$. You may see the basic representation of the system is given below in Figure (2.5).

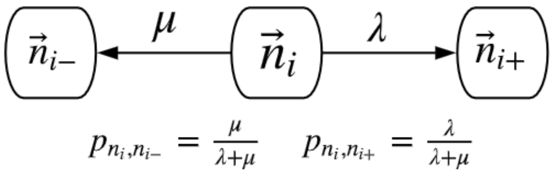
$\vec{n}_{i-}$
μ
$\vec{n}_i$
λ
$\vec{n}_{i+}$
$p_{n_i,n_{i-}} = \frac{\mu}{\lambda+\mu}$
$p_{n_i,n_{i+}} = \frac{\lambda}{\lambda+\mu}$

Figure 2.5: Transitions between states in a M/M/1 queue

You may see all the possible transitions if we have a 3 node network given in the Figure 2.6.

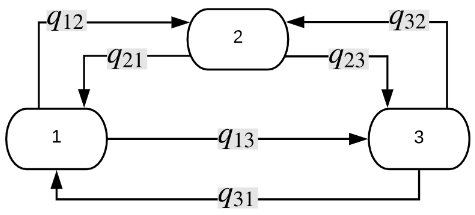

Figure 2.6: Possible transitions between nodes

In queuing networks we can explain the states of the nodes as the number of entities in each node. Therefore, if we have K nodes we can show the current state as $\overrightarrow{n} = (n_1, n_2, n_3, ..., n_i, ..., n_j, ..., n_K)$. While finding the transition probabilities, we assume that only one transition can occur at a time. It can be either one external arrival to one of the nodes, one departure from a node or it can be one entity switching nodes. Therefore if we have a 2 node queuing network we cannot have an external arrival to first node and a departure from the second node at the same time; the times where transitions occur must have a slight difference.

Let i and j be two nodes which have a connection(routing probability different than 0) between. We can show the state $\vec{n}$ where,

• One external arrival to node i occurred as:

$\overrightarrow{n_{i+1,j}} = (n_1, n_2, n_3, ..., n_i + 1, ..., n_j, ..., n_K)$

• One departure from the node i occurred as:

$\overrightarrow{n_{i-1,j}} = (n_1, n_2, n_3, ..., n_i - 1, ..., n_j, ..., n_K)$.

• One customer departed from node i and arrived to node j as:

$\overrightarrow{n_{i-1,j+1}} = (n_1, n_2, n_3, ..., n_i - 1, ..., n_j + 1, ..., n_K)$.

• One customer departed from node j and arrived to node i as:

$\overrightarrow{n_{i+1,j-1}} = (n_1, n_2, n_3, ..., n_i + 1, ..., n_j - 1, ..., n_K)$.

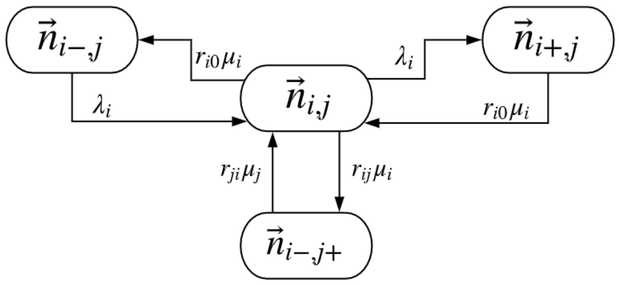

Figure 2.7: Transition rates from states

You may see the transition rates from one state to another in Figure 2.7. If we say that our initial state is $\vec{n}_{i,j}$, going to $\vec{n}_{i+,j}$ state means that one external arrival occurred with rate λ_i. If we have one departure from state $\vec{n}_{i,j}$ we will reach to state $\vec{n}_{i-,j}$ with the rate of $r_{i0}\mu_i$, where r_{i0} is the routing probability of departing from the system from node i.

There are two more options that may occur in a queuing network: a customer may leave one node and enter another node. If we are currently at the state $\vec{n}_{i,j}$ and we go to state $\vec{n}_{i-,j+}$ it means that a customer left node i and entered node j. The rate of this transition is $r_{ij}\mu_i$ where r_{ij} is the routing probability from node i to node j.

Therefore we can find the transition rates by looking at Figure 2.7. If we would like to show it mathematically, we can use $q_{\vec{n},\vec{m}}$ as the transition rates. In this representation our initial state is shown as $\vec{n}$ which represents $\vec{n}_{i,j}$, and $\vec{m}$ shows the next state where a transition occurs. You may see the open form below in equation (10).

$$q_{\boldsymbol{\vec{n}},\boldsymbol{\vec{m}}} = \begin{cases} 0, \quad \vec{m} \notin \{\vec{n}_{i+}, \vec{n}_{i-}, \vec{n}_{i+,j-}, \vec{n}_{i-,j+}\} \\ \lambda_i, \quad \vec{m} = \vec{n}_{i+,j} \\ r_{i0}\mu_i, \quad \vec{m} = \vec{n}_{i-,j} \\ r_{ji}\mu_j, \quad \vec{m} = \vec{n}_{i+,j-} \\ r_{ij}\mu_i, \quad \vec{m} = \vec{n}_{i-,j+} \\ -[\lambda_i + \mu_i + r_{ji}\mu_j + r_{ij}\mu_i], \quad \vec{m} = \vec{n} \end{cases} \tag{10}$$

Transition probabilities can be found by using the transition rates given in equation (10) and you may see the probabilities below in equation (11) shown as $p_{\boldsymbol{\vec{n}},\boldsymbol{\vec{m}}}$. In order to find the probabilities we first found all the possible transitions occur in each state by summing the rates over i and we obtained: $\phi_i = \lambda_i + r_{i0}\mu_i + r_{ji}\mu_j + r_{ij}\mu_i$.

$$p_{\boldsymbol{\vec{n}},\boldsymbol{\vec{m}}} = \begin{cases} 0, \quad \vec{m} \notin \{\vec{n}_{i+}, \vec{n}_{i-}, \vec{n}_{i+,j-}, \vec{n}_{i-,j+}\} \\ \dfrac{\lambda_i}{\phi_i}, \quad \vec{m} = \vec{n}_{i+,j} \\ \dfrac{\mu_i}{\phi_i}, \quad \vec{m} = \vec{n}_{i-,j} \\ \dfrac{r_{ji}\mu_j}{\phi_i}, \quad \vec{m} = \vec{n}_{i+,j-} \\ \dfrac{r_{ij}\mu_i}{\phi_i}, \quad \vec{m} = \vec{n}_{i-,j+} \\ 0, \quad \vec{m} = \vec{n} \end{cases} \tag{11}$$

If we have total of O states in our system, that means we have O number of $\vec{n}$ vectors and also we have O number of $\vec{m}$'s. In order to obtain the transition probability matrix we will need $O^2 - O$ number of $p_{\boldsymbol{\vec{n}},\boldsymbol{\vec{m}}}$'s where $\vec{n}, \vec{m} = 1, 2, 3, ...K$ because the diagonal element of this transition matrix is going to be equal to 0, since there is no transition from state $\vec{n}_{i,j}$ to same state. Therefore if we call our transition rate matrix as Q, our transition probability matrix is going to look like T which you can see below.

$$Q = \begin{bmatrix} -\sum_j q_{1j} & q_{12} & q_{13} & \cdots & q_{1N} \\ q_{21} & -\sum_j q_{2j} & q_{23} & \cdots & q_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & q_{N3} & \cdots & -\sum_j q_{Nj} \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & \frac{q_{12}}{\sum_j q_{1j}} & \frac{q_{13}}{\sum_j q_{1j}} & \cdots & \frac{q_{1N}}{\sum_j q_{1j}} \\ \frac{q_{21}}{\sum_j q_{2j}} & 0 & \frac{q_{31}}{\sum_j q_{2j}} & \cdots & \frac{q_{2N}}{\sum_j q_{2j}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{q_{N1}}{\sum q_{Nj}} & \frac{q_{N2}}{\sum_j q_{Nj}} & \frac{q_{N3}}{\sum_j q_{Nj}} & \cdots & 0 \end{bmatrix}$$

2.4 Closed Jackson Networks

In closed Jackson networks there is no arrival or departure from the system, basically, $\gamma_i = 0$ and $r_{i0} = 0$. That means, we have N customers continuously travelling in the system. This model is a special form of general Jackson networks. Therefore we're mainly going to use the product form solution and the formulas we have previously mentioned in open Jackson networks. After analysing the traffic equations and using "input = output" relationship.

2.5 Basic Steady State Properties of Queuing Networks

In Jackson networks we can assume every node as an individual queue, that's why in order to find steady state properties we need to find the properties of these individual queues.

In our case, we assume that the service rate of the customers are exponentially distributed and the arrival rate distribution is Poisson. We may have one or more servers in a node and if we are using Kendall's notation, we are either working with an M/M/1 queue or an M/M/c queue. The customers are coming from a infinite population.

Generally, the properties we want to know in a queue are:

- Average waiting time in the queue (W_q)
- Average waiting time in the system (W)
- Average number of people waiting in the queue (L_q)
- Average number of people in the system (L)
- Server utilization (ρ)

2.5.1 Single Server and Infinite Capacity Queue M/M/1

Arrivals are Poisson with the rate λ and service times are exponentially distributed with rate μ in this case. If the λ/μ is less than 1 the system will function properly and will not explode. In steady state we can find the properties are given above: W_q, W, L, L_q, ρ.

Utilization of the server is given as:

$$\rho = \lambda/\mu \tag{12}$$

The probability of having "n" customers in the system is shown as P_n and it is equal to:

$$p_n = (1-\rho)\rho^n \text{ and, } p_0 = 1-\rho \tag{13}$$

The average number of customers in the system and in the queue is given as:

$$L = \rho/(1-\rho) \text{ and, } L_q = \rho^2/(1-\rho) \tag{14}$$

Average waiting time in the queue and in the system is given as:

$$W = 1/(\mu-\lambda) \text{ and, } W_q = \lambda/\mu(\mu-\lambda) \tag{15}$$

2.6 Network Structure

The network that we have examined in this report is given below in Figure 2.8. It consists of 6 nodes where 2 of them are parallel to each other and the rest is in series. We only let external arrivals to first node and entities can only depart the system from the last node. Routing probability of going from the second node to third is 0.6 and going from the second node to fourth is 0.4. Rest of the routing probabilities are equal to 1, since they are connected in series.

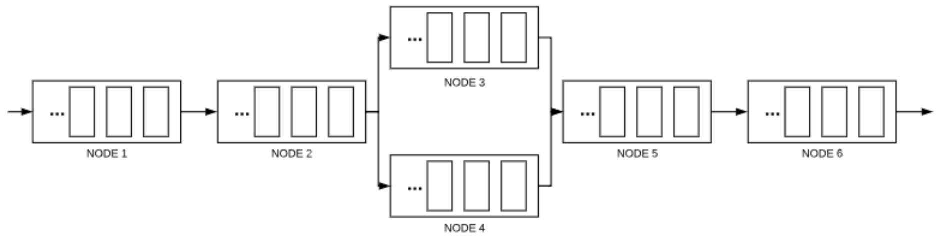

Figure 2.8: Sample open Jackson queuing network

All of the queues have only one server and we assume that their service rates are independent from each other. Their service rates are exponentially distributed with the mean μ_i and the arrivals are with Poisson distribution with rate λ_i where $i \in \{0, 1, 2, 3, 4, 5, 6\}$. According to the given information, we can say that we are working on a queueing network with 6 M/M/1 queues. While showing the general cases in other chapters we took the maximum number of queues in a network as K and the maximum observations that we have seen from a queue as O.

CHAPTER 3: LIKELIHOOD ESTIMATION

In the network model, stochasticity of the environment is often represented by a parametric probability density function. Observations have varying values how likely they are coming from different parameters for the same distribution. This value of a particular data for the parametrized density is called the the likelihood of the variable under the distribution. In statistics, likelihood of the data set under the distribution is used to measure the goodness of the fit. Hence, to compare the performance of two different parameter set for the same distribution family, likelihood ratio is often preferred. This comparison is referred as likelihood ratio test. Moreover, by nature, the logarithm is a monotonically increasing function and hence allows the likelihood ratio to be valid after applied. In most scenarios, distributions are selected from exponential families, thus taking the logarithm eases computation. For our work, we also follow the log-likelihood ratio test to justify the performance of our model.

3.1 Maximum Likelihood Estimation for Markov Chains

Generally in queuing theory states of the Markov Chain defined as the number of entities in the queue. Therefore if we have a network of K nodes and if the maximum number of states is O; our state space is going to consist O number of 1 x K sized vectors.

3.1.1 Derivation of Maximum Likelihood Estimation for Markov Chains

Let's assume that we are working with a network consists of O states and we would like to find the transient matrix, we can use maximum likelihood estimation to obtain its best estimate. Let's say that $\vec{n}_t^s$ is the O sized vector which shows the number of people in a queue at time t, and N_t^s is the realization of n_t^s and it is also an O sized vector. By looking at this information and using the basic knowledge of Markov Chains [19] we can obtain the Equation (16).

$$\begin{aligned} P(\vec{n}_1^s = \vec{N}_1^s) &= P(\vec{n}_1 = \vec{N}_1) \prod_{t=2}^{O} P(\vec{n}_t = \vec{N}_t | \vec{n}_{t-1} = \vec{N}_{t-1}) \\ &= P(\vec{n}_1 = \vec{N}_1) \prod_{t=2}^{O} p_{\vec{n}_{t-1},\vec{n}_t} \\ &= P(\vec{n}_1 = \vec{N}_1) \prod_{\vec{n}=1}^{O} \prod_{\vec{m}=1}^{O} p_{\vec{n},\vec{m}}^{c_{\vec{n},\vec{m}}} \end{aligned} \tag{16}$$

To find the likelihood we are going to make Equation (16) equal to $L(p)$. Before taking the derivative of this $L(p)$ function we are going to find the logarithm likelihood to ease the derivation step.

$$logL(p) = logP(X_1 = x_1) + \sum_{\vec{n},\vec{m}}^{L} c_{\vec{n},\vec{m}} \; logp_{\vec{n},\vec{m}} \tag{17}$$

In order to take the derivative with respect to $p_{\vec{n},\vec{m}}$ we need to use the Lagrange multipliers, you may see the steps we used below in Equation (18) and you may also look for the explanation of Lagrange multipliers from Boas's book [13]. We used γ instead of λ as our Lagrange multiplier to not cause confusions.

$$\begin{aligned} \max\ L(\gamma) &= \sum_{\vec{n}} [\sum_{\vec{m}} c_{\vec{n},\vec{m}} logp_{\vec{n},\vec{m}} - \gamma_{\vec{n}} \sum_{\vec{m}} p_{\vec{n},\vec{m}} - 1] \\ \frac{\partial L}{\partial p_{\vec{n},\vec{m}}} = \frac{c_{\vec{n},\vec{m}}}{p_{\vec{n},\vec{m}}} - \gamma_{\vec{m}} \text{ when we make } \frac{\partial L}{\partial p_{\vec{n},\vec{m}}} &= 0 \text{ we will get the maximum point} \\ p_{\vec{n},\vec{m}} = \frac{c_{\vec{n},\vec{m}}}{\gamma_{\vec{m}}} \text{ and if we sum } p_{\vec{n},\vec{m}} \text{ over} \vec{m} \text{ , we will get} \sum_{\vec{m}} p_{\vec{n},\vec{m}} &= \frac{\sum_{\vec{m}} c_{\vec{n},\vec{m}}}{\gamma_{vecm}} = 1 \\ p_{\vec{n},\vec{m}} &= \frac{c_{\vec{n},\vec{m}}}{\sum_{m}^{m} c_{\vec{n},\vec{m}}} \end{aligned} \tag{18}$$

After taking the logarithm of the likelihood and using the Lagrange multipliers we obtained that the maximum likelihood of the transition matrix can be found by using:

$$p_{\vec{n},\vec{m}} = \frac{\text{number of transactions from state } \vec{n} \text{ to state } \vec{m}}{\text{number of transactions from state } \vec{n} \text{ to all other states}}$$

3.1.2 Implementation of MLE to Transition Probabilities

Let's assume that we have a queuing network consist of K nodes and the maximum number of states is L. States are the number of entities in each queue. Therefore we have O number of 1 x K sized $\vec{n} = [n_1, n_2, ...n_K]$ vectors. If we are talking about the most general case, where there can be an arrival to each node and there can be a departure from each node we need routing probabilities between nodes and the routing probability of departing from that specific node. You may see the routing probabilities, arrival and departure rates for a 3 node network below in Figure 3.1.

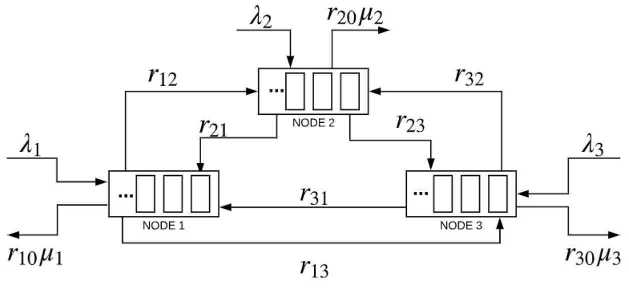

Figure 3.1: Transition rates from states

For instance if we use this network, possible consecutive states may be: $\vec{n}_1 = (0, 0, 0)$, $\vec{n}_2 = (1, 0, 0)$, $\vec{n}_3 = (0, 0, 0)$, $\vec{n}_4 = (1, 0, 0)$, $\vec{n}_5 = (2, 0, 0)$, $\vec{n}_6 = (1, 1, 0)$, $\vec{n}_7 = (1, 0, 1)$, $\vec{n}_8 = (2, 0, 1)$, $\vec{n}_9 = (1, 1, 1)$... . We can obtain the probability to maintain this sequence by finding the $P(\vec{n}_1 \rightarrow \vec{n}_2)P(\vec{n}_2 \rightarrow \vec{n}_3)P(\vec{n}_3 \rightarrow \vec{n}_4)...P(\vec{n}_{l-1} \rightarrow \vec{n}_l)$. By using the information we got from the transition probabilities part and derivation of MLE, we can write these probabilities for this network.

$$
\begin{aligned}
p(\vec{n}_1 \to \vec{n}_2) &= \frac{\lambda_1}{\lambda_1 + \mu_1 r_{12} + \mu_1 r_{13} + \mu_2 r_{21} + \mu_3 r_{31} + \mu_1 r_{10}} \\
p(\vec{n}_2 \to \vec{n}_3) &= \frac{\mu_1}{\lambda_1 + \mu_1 r_{12} + \mu_1 r_{13} + \mu_2 r_{21} + \mu_3 r_{31} + \mu_1 r_{10}} \\
p(\vec{n}_3 \to \vec{n}_4) &= \frac{\lambda_1}{\lambda_1 + \mu_1 r_{12} + \mu_1 r_{13} + \mu_2 r_{21} + \mu_3 r_{31} + \mu_1 r_{10}} \\
p(\vec{n}_4 \to \vec{n}_5) &= \frac{\lambda_1}{\lambda_1 + \mu_1 r_{12} + \mu_1 r_{13} + \mu_2 r_{21} + \mu_3 r_{31} + \mu_1 r_{10}} \\
p(\vec{n}_5 \to \vec{n}_6) &= \frac{r_{12}\mu_1}{\lambda_1 + \mu_1 r_{12} + \mu_1 r_{13} + \mu_2 r_{21} + \mu_3 r_{31} + \mu_1 r_{10}} \\
p(\vec{n}_6 \to \vec{n}_7) &= \frac{r_{23}\mu_2}{\lambda_2 + \mu_1 r_{12} + \mu_3 r_{32} + \mu_2 r_{21} + \mu_2 r_{23} + \mu_2 r_{20}} \\
&\vdots
\end{aligned} \tag{19}
$$

3.2 CUSUM Charts and Statistics

Cumulative Sum control charts introduced by Page(1958) as a memory control chart to detect changes in processes faster [26]. Currently, they are known to be giving satisfactory results therefore, they are used as a statistical process control tool for monitoring processes. In our study, we also used CUSUM charts instead of Shewhart charts because while Shewhart charts don't keep previous data, CUSUM charts fall into the memory-type control charts [30]. And the most popular performance measure of these charts is the average run length (ARL). Addition to Page(1958)'s study and Faisal(2018)'s study we learned from Basseville(1993) that we can derive CUSUM charts by using likelihood ratios [27].

The ARL_0 represents the average run length when the system is in control and ARL_1 represents the average run length when the system is out of control. Therefore if the system is in control we want ARL_0 to be high and when the system is out of control we want ARL_1 to be low as possible.

If we call $\mathbf{\Theta}_0 = [\mu_1, \mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, r_{12}, r_{23}..]$ as our in control parameters which represents service rates, arrival rates and routing probabilities, $\mathbf{\Theta}_1 = [(1+\delta_1)\mu_1, (1+$

$\delta_1)\mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, r_{12}, r_{23}..]$ will be our design parameters which also represents the same properties as in control parameters. And let $\vec{n}_k = [n_1, n_2, ...n_K]$ be our kth observation from queueing network where all elements of the vector represents the number of entities in queues of the network. We can represent our in control observations as joint probability density function(pdf) as $P(\vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_O|\mathbf{\Theta_0})$ and designed pdf as $P(\vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_O|\mathbf{\Theta_1})$. With the help of these joint pdf's we can find the likelihood and log-likelihood ratios which will help us construct the CUSUM chart as follows:

$$\frac{P(\vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_O|\mathbf{\Theta_1})}{P(\vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_O|\mathbf{\Theta_0})} \quad \text{we can also obtain the log likelihoods by,}$$

$$logP(\vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_O|\mathbf{\Theta_1}) - logP(\vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_O|\mathbf{\Theta_0})$$

We can obtain CUSUM statistics by using conditional probability rule and the log likelihoods that we have found. Conditional probability rule is given:

$$P(\vec{n}_k|\vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1, \mathbf{\Theta}) = \frac{P(\vec{n}_k, \vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1|\mathbf{\Theta})}{P(\vec{n}_{k-1}, \vec{n}_{k-2}, ...\vec{n}_2, \vec{n}_1|\mathbf{\Theta})}$$

By applying the conditional probability rule we get,

$$logP(\vec{n}_k|\vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1, \mathbf{\Theta_1}) = log\frac{P(\vec{n}_k, \vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1|\mathbf{\Theta_1})}{P(\vec{n}_{k-1}, \vec{n}_{k-2}, ...\vec{n}_2, \vec{n}_1|\mathbf{\Theta_1})}$$

$$logP(\vec{n}_k|\vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1, \mathbf{\Theta_0}) = log\frac{P(\vec{n}_k, \vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1|\mathbf{\Theta_0})}{P(\vec{n}_{k-1}, \vec{n}_{k-2}, ...\vec{n}_2, \vec{n}_1|\mathbf{\Theta_0})}$$

Let CUSUM statistic be g_k where $g_0 = 0$,

$$g_k = \max\{0, g_{k-1} + (logP(\vec{n}_k|\vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1, \mathbf{\Theta_1}) \; - \; logP(\vec{n}_k|\vec{n}_{k-1}, ...\vec{n}_2, \vec{n}_1, \mathbf{\Theta_0}))\}$$

If we call these log likelihood functions as $l_k(\mathbf{\Theta_1})$ and $l_k(\mathbf{\Theta_0})$ we can call the difference as $\tau_k = l_k(\mathbf{\Theta}_1) - l_k(\mathbf{\Theta}_0)$ and this will make our CUSUM statistic equal to $g_k = \max\{0, g_{k+1} + \tau_k\}$.

3.3 Likelihood Ratio Tests

If we have two communicating states named $\vec{n}$ and $\vec{m}$ we can represent them as shown below.

$$\vec{n} = [n_1, n_2, ...n_K]$$

$$\vec{m} = [m_1, m_2, ...m_K]$$

One step transitions between these states were explained before, in section 4.2 Transition Probabilities. Therefore we know that we have 4 possible transitions: an arrival, a departure or a switch between nodes. In order to write the transition probabilities in general form we need the differences between states and we defined these vectors as $\vec{e}_{i+}$, $\vec{e}_{i-}$, $\vec{e}_{i+j-}$, $\vec{e}_{i-j+}$.

$$\vec{e}_{i+} = \begin{bmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{bmatrix} \quad \text{for the } i^{th} \quad \text{component}$$

$$\vec{e}_{i-} = \begin{bmatrix} 0 & 0 & \dots & -1 & \dots & 0 \end{bmatrix} \quad \text{for the } i^{th} \quad \text{component}$$

$$\vec{e}_{i+j-} = \begin{bmatrix} 0 & \dots & 1 & \dots & -1 & \dots & 0 \end{bmatrix} \quad \text{for the } i, j \quad \text{components}$$

$$\vec{e}_{i-j+} = \begin{bmatrix} 0 & \dots & -1 & \dots & 1 & \dots & 0 \end{bmatrix} \quad \text{for the } i, j \quad \text{components}$$

All the possible states that a specific state can go defined as:

$$\phi_i = \lambda_i + (1 - \sum_{j=1, j \neq i}^{K} r_{ji})\mu_i + \sum_{j=1, j \neq i}^{K} r_{ij}\mu_i + \sum_{j=1, j \neq i}^{K} r_{ji}\mu_j$$

We can show the departures from the system by using:

$$r_{i0} = (1 - \sum_{j=1, j \neq i}^{K} r_{ij})$$

Probability of going from state $\vec{n}$ to state $\vec{m}$ can be shown below. We used the Equation (11) from section 4.2 Transition Probabilities to find this definition.

$$p_{\vec{\boldsymbol{n}},\vec{\boldsymbol{m}}} = \begin{cases} 0 & \text{if} \quad \vec{\boldsymbol{n}} - \vec{\boldsymbol{m}} \neq \{\mathbf{0}, \vec{\boldsymbol{e}_{i+}}, \vec{\boldsymbol{e}_{i-}}, \overrightarrow{\boldsymbol{e}_{i+j-}}\} \\ \dfrac{\lambda_i}{\phi_i} & \text{if} \quad \vec{\boldsymbol{n}} - \vec{\boldsymbol{m}} = \vec{\boldsymbol{e}_{i+}} \\ \dfrac{r_{i0}\mu_i}{\phi_i} & \text{if} \quad \vec{\boldsymbol{n}} - \vec{\boldsymbol{m}} = \vec{\boldsymbol{e}_{i-}} \\ \dfrac{r_{ij}\mu_i}{\phi_i} & \text{if} \quad \vec{\boldsymbol{n}} - \vec{\boldsymbol{m}} = \overrightarrow{\boldsymbol{e}_{i-,j+}} \\ \dfrac{r_{ji}\mu_j}{\phi_i} & \text{if} \quad \vec{\boldsymbol{n}} - \vec{\boldsymbol{m}} = \overrightarrow{\boldsymbol{e}_{i+,j-}} \end{cases} \tag{20}$$

From a queuing network we observe O number of observations which are shown as:

$$\vec{\boldsymbol{n}}_0, \vec{\boldsymbol{n}}_1, \ldots \vec{\boldsymbol{n}}_O$$

Define,

$$c_{\vec{\boldsymbol{n}},\vec{\boldsymbol{m}}} = \sum_{i=1}^{O-1} \mathbb{I}\left(\vec{\boldsymbol{n}_i} = \vec{\boldsymbol{n}} \text{ and } \vec{\boldsymbol{n}_{i+1}} = \vec{\boldsymbol{m}}\right)$$

$c_{\vec{\boldsymbol{n}},\vec{\boldsymbol{m}}}$ is going to help us count the number of $\vec{e}_{i+}, \vec{e}_{i-}, \vec{e}_{i-,j+}, \vec{e}_{i+,j-}$ vectors we find after each transition.

Likelihood function obtained as follows:

$$\begin{aligned} & L(\mu_1, \mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, R) \\ =& \prod_{i=1}^{K} \prod_{j=1, j\neq i}^{K} (p_{\boldsymbol{e}_{i+}})^{c_{\boldsymbol{e}_{i+}}} (p_{\boldsymbol{e}_{i-}})^{c_{\boldsymbol{e}_{i-}}} (p_{\boldsymbol{e}_{i+j-}})^{c_{\boldsymbol{e}_{i+j-}}} (p_{\boldsymbol{e}_{i-j+}})^{c_{\boldsymbol{e}_{i-j+}}} \\ =& \prod_{i=1}^{K} \prod_{j=1, j\neq i}^{K} \left(\frac{\lambda_i}{\phi_i}\right)^{c_{\boldsymbol{e}_{i+}}} \left(\frac{r_{i0}\mu_i}{\phi_i}\right)^{c_{\boldsymbol{e}_{i-}}} \left(\frac{r_{ji}\mu_j}{\phi_i}\right)^{c_{\boldsymbol{e}_{i+j-}}} \left(\frac{r_{ij}\mu_i}{\phi_i}\right)^{c_{\boldsymbol{e}_{i-j+}}} \end{aligned} \tag{21}$$

Log-likelihood function obtained by taking logarithm of likelihood function (Equation (21)) and it's given below.

$$
\begin{aligned}
&l_O(\mu_1, \mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, R) \\
=&\sum_{i=1}^{K} \sum_{j=1, j\neq i}^{K} c_{\boldsymbol{e}_{i+}} \log\left(\frac{\lambda_i}{\phi_i}\right) + c_{\boldsymbol{e}_{i-}} \log\left(\frac{r_{i0}\mu_i}{\phi_i}\right) + c_{\boldsymbol{e}_{i+j-}} \log\left(\frac{r_{ji}\mu_j}{\phi_i}\right) + c_{\boldsymbol{e}_{i-j+}} \log\left(\frac{r_{ij}\mu_i}{\phi_i}\right)
\end{aligned}
\tag{22}
$$

Let,

$$\boldsymbol{\Theta}_0 = [\mu_1, \mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, r_{12}r_{23}..]$$

$$\boldsymbol{\Theta}_1 = [(1+\delta_1)\mu_1, (1+\delta_1)\mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, r_{12}r_{23}..]$$

where Θ_0 represents the in control parameters and Θ_1 represents the design parameters. By using the likelihood ratios we obtained,

$$\tau_k = l_k(\boldsymbol{\Theta}_1) - l_k(\boldsymbol{\Theta}_0)$$

Which helped us find the CUSUM statistic is defined as:

$$g_k = \max\{0, g_{k-1} + \tau_k\} \tag{23}$$

Monitoring rule if,

$$g_k \geq h$$

we say that the system has gone slow. In this case h is the threshold value and also it can be interpreted as Upper Control Limit(UCL). When a point above that threshold value observed, we say that the process is out of control. Since that point is the first out of control point that we have observed, it can also be called as ARL.

CHAPTER 4: SIMULATION STRUCTURE AND RESULTS

4.1 Structure of the Simulation

Aim of this simulation is to detect the changes in service rates in queueing networks by looking at the queue lengths. To detect changes faster we used CUSUM charts derived from likelihood ratios and we used the ARL values to compare performances of different design parameters.

Initial values that we used is $\vec{\mu}_0 = [1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$ as our service rate and arrival rate as $\vec{\lambda} = [1, 1, 1, 1, 1, 1]$. Our routing matrix is given below.

$$\text{Routing Matrix } (R) = \begin{bmatrix} 0 & 1 & 0.0 & 0.0 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0.0 & 0.0 & 1 & 0 \\ 0 & 0 & 0.0 & 0.0 & 1 & 0 \\ 0 & 0 & 0.0 & 0.0 & 0 & 1 \\ 0 & 0 & 0.0 & 0.0 & 0 & 0 \end{bmatrix}$$

With all the initial values we defined our in control parameters which is defined before as $\mathbf{\Theta}_0$,you may see below.

$$\mathbf{\Theta}_0 = [\mu_1, \mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, R] = [\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$$

By just changing service rates:

$$\text{from } \vec{\mu}_0 = [1.1, 1.1, 1.1, 1.1, 1.1, 1.1] \text{ to } \begin{cases} \vec{\mu}_1 = 0.9[1.1, 1.1, 1.1, 1.1, 1.1, 1.1] \\ \vec{\mu}_2 = [0.9(1.1), 1.1, 1.1, 1.1, 1.1, 1.1] \end{cases}$$

we obtained our design parameters $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ as shown below. We only changed the service rates and kept all the other parameters as they are because in this simulation we only focused on effect of change in service rates in queuing networks.

$$\mathbf{\Theta}_1 = [(1+\delta_1)\mu_1, (1+\delta_1)\mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, r_{12}r_{23}..] =$$

$$[\vec{\mu} = 0.9[1.1, 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$$

$$\mathbf{\Theta}_2 = [(1+\delta_1)\mu_1, (1+\delta_1)\mu_2, ...\mu_K, \lambda_1, \lambda_2, ...\lambda_K, r_{12}r_{23}..] =$$

$$[\vec{\mu} = \vec{\mu}_2 = [0.9(1.1), 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$$

By using these in control and design parameters we calculated the likelihood values which then helped us draw the CUSUM charts. We had 2 layouts of CUSUM charts for finding ARL$_0$: (i)$\mathbf{\Theta}_1$ and $\mathbf{\Theta}_0$,(ii) $\mathbf{\Theta}_2$ and , $\mathbf{\Theta}_0$. After plotting the CUSUM charts we tried finding the threshold value(h) which will give us the ARL$_0$ equal to 100. Our replication length was 10000 during these processes. You may see the Table 1 to see the change of ARL$_0$ with different threshold values for $\mathbf{\Theta}_1$.

Table 1: Table of average run length values and respective threshold values

Trials	Threshold value(h)	ARL	Trials	Threshold value(h)	ARL
1	0.30000	91.6063	13	0.31800	100.7563
2	0.31200	98.6633	14	0.31805	**100.0016**
3	0.31300	98.1186	15	0.31810	99.6200
4	0.31350	98.6633	16	0.31820	100.3964
5	0.31500	101.2301	17	0.31830	100.3017
6	0.31600	98.2884	18	0.31840	99.0278
7	0.31700	101.0299	19	0.31850	100.2106
8	0.31750	99.8485	20	0.31900	101.4304
9	0.31770	100.1145	21	0.31900	101.8636
10	0.31780	98.7236	22	0.32000	101.0877
11	0.31790	102.0267	23	0.33000	108.7100
12	0.31800	99.7828	24	0.40000	160.0700

So by looking at the table we took our threshold value(h) to be 0.318 for the first CUSUM chart. For the second CUSUM chart we found the threshold value as 0.345. In addition to these we simulated the process by using different design parameters, in every 10000 replication we changed our service rates and found the corresponding ARL values for all CUSUM charts. Since our aim is to spot the time where the service become slower, we decreased our service rates and checked whether the ARL values are decreasing or not. You may see the list of design parameters below in Table 2. We worked on these parameters with 2 different CUSUM layouts that we have, and compared all the ARL values. Average Run Length is the expected number observations before an out of control point observed. That's why if our system is out of control we want ARL value to be small as possible and if the system is in control we want ARL value to be high as possible. In our case, we want ARL value to decrease whenever the service rate is decreasing. Because slow servers mean that the system is getting out of control and we would like to spot the out of control signal fast as possible to find out the cause and solve the problem earlier.

Table 2: Different design parameters that we have used while simulation

Explanations	Design of service parameters
$\vec{\mu}$ changed (all service rates changed)	$\vec{\mu} = 0.9[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = 0.8[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = 0.7[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = 0.6[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = 0.5[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = 0.4[1.1, 1.1, 1.1, 1.1, 1.1, 1.1]$
First element of $\vec{\mu}$ changed	$\vec{\mu} = [0.9(1.1), 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = [0.9(1.1), 1.1, 1.1, 1.1, 1.1, 1.1]$
	$\vdots$
	$\vec{\mu} = [0.9(1.1), 1.1, 1.1, 1.1, 1.1, 1.1]$
Second element of $\vec{\mu}$ changed	$\vec{\mu} = [1.1, 0.9(1.1), 1.1, 1.1, 1.1, 1.1]$
	$\vec{\mu} = [1.1, 0.8(1.1), 1.1, 1.1, 1.1, 1.1]$
	$\vdots$
	$\vec{\mu} = [1.1, 0.4(1.1), 1.1, 1.1, 1.1, 1.1]$
Third element of $\vec{\mu}$ changed	$\vec{\mu} = [1.1, 1.1, 0.9(1.1), 1.1, 1.1, 1.1]$
	$\vec{\mu} = [1.1, 1.1, 0.8(1.1), 1.1, 1.1, 1.1]$
	$\vdots$
	$\vec{\mu} = [1.1, 1.1, 0.4(1.1), 1.1, 1.1, 1.1]$
Fifth element of $\vec{\mu}$ changed	$\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 0.9(1.1), 1.1]$
	$\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 0.8(1.1), 1.1]$
	$\vdots$
	$\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 0.4(1.1), 1.1]$
Sixth element of $\vec{\mu}$ changed	$\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 1.1, 0.9(1.1)]$
	$\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 1.1, 0.8(1.1)]$
	$\vdots$
	$\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 1.1, 0.4(1.1)]$

4.2 Simulation Results

In simulation we used the design parameters that are shown in Table 2 with 2 different CUSUM layouts which named as CUSUM$_1$ and CUSUM$_2$.

CUSUM$_1$ corresponds to using:

$\mathbf{\Theta}_0 = [\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$ and

$\mathbf{\Theta}_1 = [\vec{\mu} = 0.9[1.1, 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$ with $h = 0.31805$

And CUSUM$_2$ corresponds to using:

$\mathbf{\Theta}_0 = [\vec{\mu} = [1.1, 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$ and

$\mathbf{\Theta}_2 = [\vec{\mu} = [0.9(1.1), 1.1, 1.1, 1.1, 1.1, 1.1], \vec{\lambda} = [1, 1, 1, 1, 1, 1], R]$ with $h = 0.345$

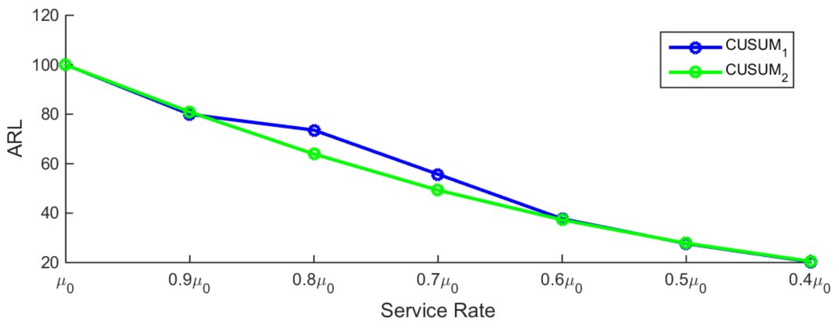

Figure 4.1: Change in ARL values when all the service rates decrease in the network

In Figure 4.1 we changed our $\vec{\mu}$ itself, that means all the servers on the network have gone slow. As you can see the Average Run Length value is decreasing when we decrease the service rates. Thus, we can say that we are detecting the out of control points faster whenever the system goes a bit slower. Both CUSUM$_1$ and CUSUM$_2$ designs seem to be giving satisfactory results. But we were expecting CUSUM$_1$ design to be giving better results compared to CUSUM$_2$ while all the servers are becoming slower. Because $\mathbf{\Theta}_1$ parameters of CUSUM$_1$ is specially designed for spotting the change while all the servers are going slow. After running our simulation we saw that CUSUM$_2$ is giving better results, thus we realized that slowing down in first queue is dominating the network. On the other hand, since the difference between results are not so much we can say that both designs work well while spotting the change in service rates.

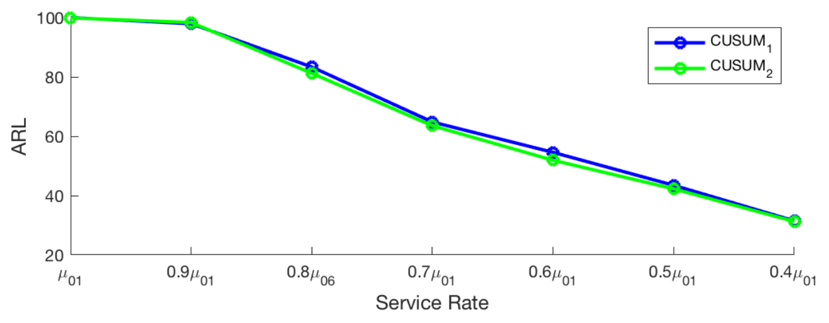

Figure 4.2: Change in ARL values when just the service rate of first queue decreases

In Figure 4.2 only the service rate of the first queue is decreasing which corresponds to μ_{01}. As you can see the ARL values are decreasing when the service rate decreases and this means that we are able to spot the changes earlier when the system is out of control with both of the CUSUM designs. While only changing μ_{01} we expect CUSUM$_2$ to operate better than CUSUM$_1$ because it's designed to spot changes better when the first server gone slow. So by looking at the Figure above, we can say that CUSUM$_2$ is giving better results than CUSUM$_1$, however, the difference is not too much. On the other hand, CUSUM$_1$ is designed to detect changes when all the queues(including first queue) slow down, therefore it's also normal to observe satisfactory results from CUSUM$_1$.

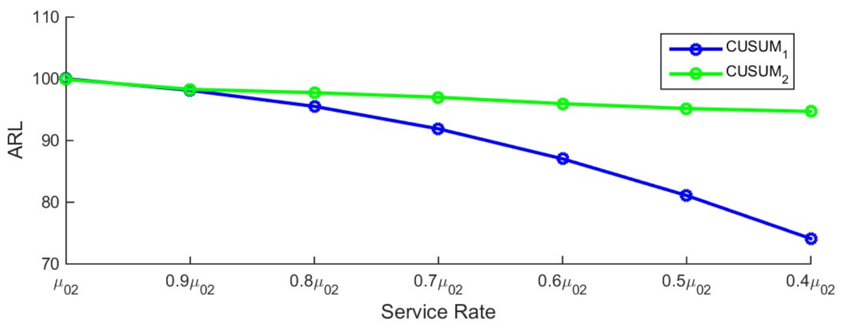

Figure 4.3: Change in ARL values when just the service rate of second queue decreases

You may see the change in ARL values in Figure 4.3 when only the service rate of the second queue decreases which is shown as μ_{02}. As you can see the CUSUM$_1$ design is giving better results compared to CUSUM$_2$. Because CUSUM$_2$ parameters don't have a decrease in the second queue's service rate and it's difficult to observe the change in the second queue. Therefore we can say that CUSUM$_1$ design is giving satisfactory results since it has a decrease in all servers when CUSUM$_2$ fails in this case.

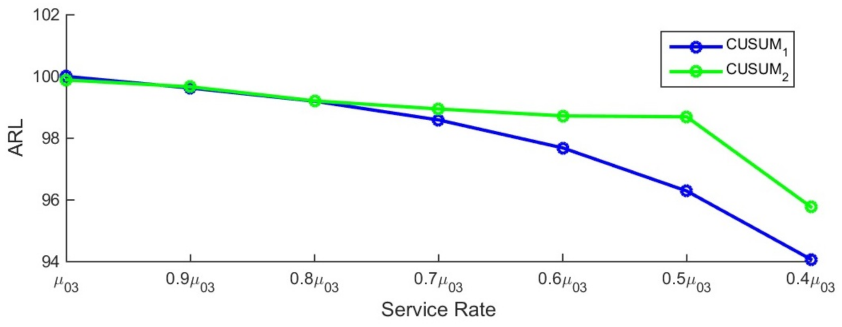

Figure 4.4: Change in ARL values when just the service rate of third queue decreases

To observe the change when only the third queue goes slow we plotted Figure 4.4. If you don't remember our network structure, third and fourth queues are parallel to each other; arrivals are coming from the second queue and the departing entities arrive in the fifth queue. As you can see from the Figure 4.4 that decrease in ARL value is not as high as the other graphs because the decrease in service rate in a parallel queue does not effect the network as much as the series queues. Therefore, in this case, ARL is decreasing from 100 to 94 with CUSUM$_1$ and from 100 to 95 with CUSUM$_2$. We can say that CUSUM$_1$ is giving more satisfactory results because in $\mathbf{\Theta}_1$ we change the third queue's service rate as well as the others. We did not work on service rates of the fourth queue since it's just parallel to the third queue and will give similar results.

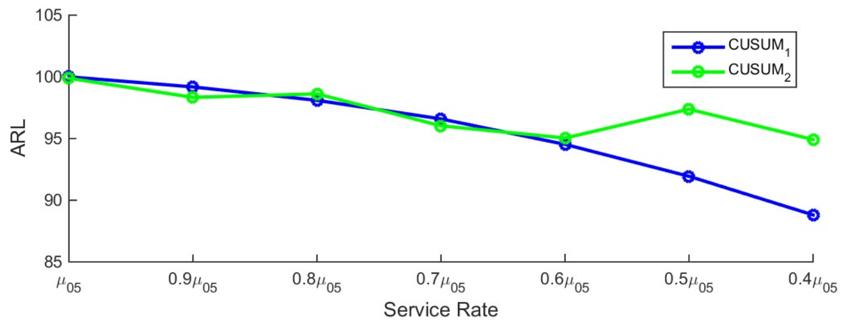

Figure 4.5: Change in ARL values when just the service rate of fifth queue decreases

You may see the change in ARL values when only the service rate of fifth queue deceases (μ_{05}) in Figure 4.5. As you can see while we can observe a steady decrease in CUSUM_1 we observe fluctuating decrease in CUSUM_2 and it's not decreasing as much as CUSUM_1. It's because CUSUM_2 doesn't have a change in fifth queue's service rates. Therefore we can say that using CUSUM_1 is more effective than using CUSUM_2 for this method.

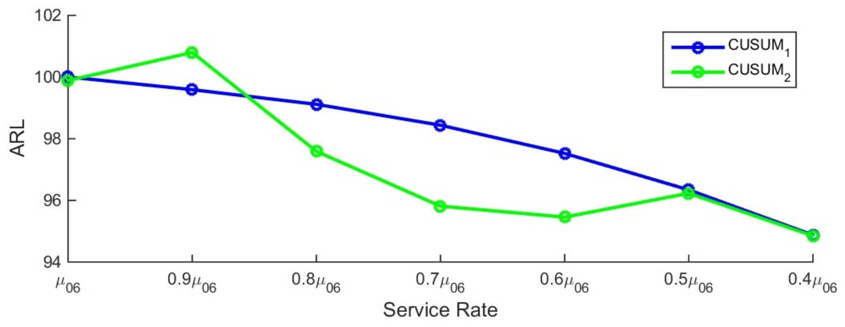

Figure 4.6: Change in ARL values when just the service rate of sixth queue decreases

In Figure 4.6 we only decreased the service rate of the sixth node (μ_{06}). The sixth node is the last node of our network, therefore, decrease in its service rates does not effect the queue lengths of the other queues in our network. Therefore we can say that the slight decrease in the ARL value has a reason. As you can see while initial and last points of CUSUM_1 and

CUSUM$_2$ are same; CUSUM$_1$ has a steady decrease and CUSUM$_2$ has fluctuations. Even though it's hard to say which design works better we can say that they are both giving satisfactory results since the ARL is decreasing.

To summarize our results, in most of the scenarios CUSUM$_1$ works better than CUSUM$_2$ while detecting change in the service rates of queues. When there is a decrease at first queue's service rate, we observed that CUSUM$_2$ is better at spotting changes compared to CUSUM$_1$. Because CUSUM$_2$'s $\boldsymbol{\Theta}_1$ parameters designed specially to detect the changes whenever first queue slows down. On the other hand, CUSUM$_1$ also gave good results for the first queue because it is prepared for detecting change in all queues. Therefore it's logical to say that by using $\boldsymbol{\Theta}_0$ and $\boldsymbol{\Theta}_1$ parameters we can detect changes faster while a random server is going slow. On the other hand, when we changed all the service rates at the same time CUSUM$_1$ should have worked better but CUSUM$_2$ gave a better result with a slight difference. Therefore we can say that they both operated well when all the service rates decreased. The main reason CUSUM$_2$ gave a better result is, first queue is carrying the biggest importance in the network since it's the source queue. In addition to these, we realized that slow down of service in parallel queues does not effect rest of the queues in the network as much as series queues. Therefore detecting the change in service rates in parallel queues is not as easy as detecting changes in series queues. Additionally, the decrease in ARL values are decreasing when we go from changing first queue's service rate to last queue's service rate. When we change the service rate of first queue the decrease in ARL is around 80 and when we change the second queue it's around 30 and when we change the third, fifth and sixth queue's service rates the ARL difference is decreasing to 5. Therefore we can say that detecting changes in first queues is easier than detecting change in last queues. In conclusion, we realized that the decrease in service rate of the last queue does not effect the queue lengths of the other queues. Therefore, both CUSUM$_1$ and CUSUM$_2$ charts are giving satisfactory results however the results are not as good as the other simulations while detecting change in last queue.

CHAPTER 5: CONCLUDING REMARKS

We proposed CUSUM charts based on likelihood ratios to monitor change in service rates of queueing networks in this report. We used a popular CUSUM performance measure: Average Run Length to compare our results. With the help of our simulations, we realized that CUSUM charts are giving satisfactory results even though the change in parameters is very small.

We mainly worked on Open Jackson Networks with M/M/1 queues. To monitor the queue lengths by using CUSUM charts based on likelihood ratios we first found the transition probabilities of queuing networks to help us find the likelihood ratios. We found the transition probabilities by looking at the difference between two consecutive states and counting the total number of times that we have seen that specific transition. By using this information, we came up with the likelihood function and took the logarithm to obtain the log-likelihood function which we used while finding the CUSUM statistic g_k. Our aim was to detect the change in service rates by monitoring queue lengths in this report and we achieved that by following this path and then we have constructed our simulation.

In our simulation, we used two different CUSUM layouts: CUSUM$_1$ was designed for detecting change whenever all the servers slow down and CUSUM$_2$ was designed for detecting change whenever only the server of the first queue slows down. After changing our design parameters by using Table 2 we draw our CUSUM charts and compared Average Run Length values of CUSUM$_1$ and CUSUM$_2$. From our simulation results, we have seen that CUSUM$_1$ and CUSUM$_2$ designs are giving satisfactory results when we decrease all the servers or when we only decrease the first server. CUSUM$_2$ gave better results in these scenarios because we realized that first queue is dominating the network. For the other scenarios, we found out that CUSUM$_1$ is giving better results compared to CUSUM$_2$. Because in CUSUM$_2$ layout we did not change the service rates of the queues except first queue. Therefore it's not sensitive to detect changes in other queues. In addition, in the scenarios where we decreased service rates of second, third, fifth and sixth queues separately we realized that ARL values are not decreasing as much as when we decreased the service rate of first queue or all the service rates at the same time. It's because of the queue length change in the first queue is making a bigger effect on the network than the other queues. Therefore while changing the service rates sequentially starting from first queue the change in ARL is decreasing. You may see from the graphs that we have plotted that the ARL value change is around 80's in first queue analysis while the change is around 5 for the last queue.

In this work we make the following contributions: (i) We monitored queuing lengths on queuing networks instead of individual queues different from previous studies (ii) We used CUSUM charts based on likelihood ratios to detect the change in service rates. (iii) We used average run length while comparing performances of different control charts. Thus, with these contributions we can say that we achieved our aim which was following and expanding previous research. The work we have done in this research is important because limited number of research conducted on the statistical monitoring of queues and especially of queueing networks.

In this study, we have only worked on networks with M/M/1 queues and we only focused on change in service rates. Therefore this study can be expanded by implementing the same theory on queueing networks with different kinds of queues such as M/G/1 and other properties of the queues may change. We can extend the work we are doing right now by using another control chart like T^2 and compare our values. The network we are using right now is a 6 node network, which is big enough to study the theory but is not considered as a complex network. In future works, the network can be extended to a much larger network. Additionally, we used a lot of parameters to come up with our model and in future, the network can be represented with lower number of parameters by using model selection problems. In addition, currently, we are working on theory and doing our own simulations based on service and arrival rates that we have decided. Therefore the most important thing to be done in future in this study is: implementing this study in a real-life system.

REFERENCES

[1] John F. Shortle, James M. Thompson, Donald Gross, Carl M. Harris. (2018) *Fundamentals of queuing Theory* John Wiley & Sons, New Jersey

[2] Nick T. Thomopoulos.(2012) *Fundamentals of Queuing Systems, Statistical Methods for Analyzing Queuing Models* Springer, Boston, MA

[3] Erol Gelenbe, Guy Pujolle. (1987) *Introduction to queuing Networks* John Wiley & Sons, New York

[4] Dimitri P. Bertsekas, Robert G. Gallager (1992) *Data Networks* Prentice Hall, Englewood Cliffs

[5] Samuel Karlin. (2014) *First Course in Stochastic Processes* Elsevier Science & Technology, ProQuest Ebook Central

[6] U. Narayan Bhat,S. Subba Rao. *A Statistical Technique for the Control of Traffic Intensity in the Queuing Systems M/G/1 and Gi/M/1* Operations Research. 20: 955-966, 1972

[7] Haim Shore. *Control Charts for the Queue Length in a G/G/S System* IIE Transactions. 38: 1117-1130, 2006

[8] David G. Kendall. *Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain* The Annals of Mathematical Statistics. 24(3):338-354,1953

[9] J.D.C. Little, S.C. Graves. (2008) *Little's Law. In:Building Intuition. International Series in Operations Research & Management Science. vol 115* Springer, Boston

[10] John D. C. Little. *Little's Law as Viewed on Its 50th Anniversary* Operations Research 59(3):536-549,2011

[11] John D. C. Little. *A Proof for the Queuing Formula:* $L = \lambda W$ Operations Research 9(3): 383-387, 1961

[12] M. Stojcic, D. Pamucar,E. Mahmutagic,Z. Stevic. *Development of an ANFIS Model for the Optimization of a Queuing System in Warehouses.* Information, 9(10): 240, 2018

[13] Mary L. Boas. (2005) *Mathematical Methods in the Physical Sciences.* John Wiley & Sons, New Jersey

[14] James R. Jackson. *Jobshop-Like queuing Systems* Management Science 50(12 supplement):1796-1802, 2004

[15] D.R. Cox, Valerie Isham. (1980) *Point Processes.Chapman & Hall/CRC Monographs on Statistics & Applied Probability. vol 12* CRC Press, Boca Raton

[16] Jim Pitman. (1993) *Probability.* Springer, New York

[17] Manish K. Govil, Michael C. Fu. *Queuing theory in manufacturing: A survey* Journal of Manufacturing Systems, 18(3):214-240, 1999.

[18] C. Lakshmi , Sivakumar Appa Iyer. *Application of queuing theory in health care: A literature review* Operations Research for Health Care, 2(1–2):25-39, March–June 2013

[19] Iuliana Teodorescu. *Maximum likelihood estimation for Markov Chains.* arXiv preprint arXiv:0905.4131. 2009 May.

[20] Douglas Montgomery. (2005) *Introduction to Statistical Quality Control* John Wiley & Sons, New Jersey

[21] K.D. Johnson, C. Winkelman. *The effect of emergency department crowding on patient outcomes: a literature review.* Advanced Emergency Nursing Journal, 33(1):39-54, 2011

[22] Paul A. Gagniuc. (2017) *Markov Chains: From Theory to Implementation and Experimentation.* John Wiley & Sons, New Jersey

[23] Nan Chen, Yuan Yuan, Shiyu Zhou. *Performance Analysis of Queue Length Monitoring of M/G/1 Systems* Naval Research Logistics, Vol. 58: 782-794 ,2011

[24] Nan Chen, Shiyu Zhou. *CUSUM Statistical Monitoring of M/M/1 Queues and Extensions* Technometrics, 57(2): 245-256, 2014

[25] Nan Chen, Shiyu Zhou. *Supplemental Materials for CUSUM Schemes for Statistical Monitoring of queuing Systems* Technometrics, 57(2): A2, 2014

[26] E. S. Page. *Continuous Inspection Schemes* Biometrika, 41(1/2): 100-115, 1954.

[27] Michele Basseville, Igor V. Nikiforov.(1993) *Detection of Abrupt Changes: Theory and Application* Prentice Hall, Englewood Cliffs

[28] Moh Zainal Arifin, Banun Diyah Probowati,Sri Hastuti. *Applications of Queuing Theory in the Tobacco Supply* Agriculture and Agricultural Science Procedia, 3: 255-261, 2015

[29] J. H. Ryu, H. Wan, S. Kim. *Optimal Design of a Cusum Chart for a Mean Shift of Unknown Size* Journal of Quality Technology, 42: 311-326, 2010

[30] Muhammad Faisal, Raja Fawad Zafar, Nasir Abbas, Muhammad Riaz, Tahir Mahmood. *A modified CUSUM control chart for monitoring industrial processes* Quality and Reliability Engineering International, 34:1045–1058, 2018

[31] J. M. Lucas. *Counted Data Cusums* Technometrics, 27: 129-144, 1985

APPENDIX A: SIMULATION MODEL

A.1 Long Form of Simulations

You may find the full form of the codes that we have used while simulating the queuing networks in this part. Explanations were in Chapter 3 and Chapter 5. The R packages that we have used are given below.

```
install.packages("Rcpp")
install.packages("RcppArmadillo")
install.packages("queuecomputer")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("tidyr")
```

Generally the initial values that we have used are below.

```
lambdas = c(1,1,1,1,1,1)
mus = c(1.1,1.1,1.1,1.1,1.1,1.1)
t_lim = 24
```

A.1.1 Simulation for Computing Queue Lengths

This code is the long form of the code that we have given in Chapter 3, which is for finding queue lengths of the queuing network and also for finding the step by step transitions. As the output, it's giving the vectors $\vec{n} = [n_1, n_2, ..., n_K]$ which shows us the transitions.

```
QL_simulator = function(lambdas, mus, t_lim) {
  ex_arr_1 = cumsum(rexp(10000,lambdas[1]))
  ex_arr_1 = ex_arr_1[ex_arr_1 <= t_lim]



  #Node 1
  arr_1 = ex_arr_1
  ser_1 = rexp(length(ex_arr_1),mus[1])
  dep_1 = queue(arr_1, ser_1,1)
  queuedata1 <- queue_lengths(arr_1, ser_1, dep_1)

  #Node 2
  arr_2=dep_1
  ser_2 = rexp(length(arr_2),mus[2])
  dep_2 = queue(arr_2, ser_2,1)
  queuedata2 <- queue_lengths(arr_2, ser_2, dep_2)

  #seperation of entities
  tmp_1 = runif(length(ex_arr_1), min=0, max=1)
  alpha = 0.6

  index1 = which(tmp_1<alpha)
  index2 = which(tmp_1> alpha)

  #Node 3
  arr_3 = dep_2[index1]
  ser_3 = rexp(length(arr_3),mus[3])
  dep_3 = queue(arr_3, ser_3,1)
  queuedata3 <- queue_lengths(arr_3, ser_3, dep_3)

  #Node 4
  arr_4 = dep_2[index2]
  ser_4 = rexp(length(arr_4),mus[4])
  dep_4 = queue(arr_4, ser_4,1)
  queuedata4 <- queue_lengths(arr_4, ser_4, dep_4)

  #Node 5
  arr_5 = sort(append(dep_3,dep_4), decreasing = FALSE)
  ser_5 = rexp(length(arr_5),mus[5])
  dep_5 = queue(arr_5, ser_5,1)
  queuedata5 <- queue_lengths(arr_5, ser_5, dep_5)

  #Node 6
  arr_6=dep_5
  ser_6 = rexp(length(arr_6),mus[6])
  dep_6 = queue(arr_6, ser_6,1)
  queuedata6 <- queue_lengths(arr_6, ser_6, dep_6)

  max.len = max(length(arr_1), length(arr_2), length(arr_3), length(arr_4),
      length(arr_5), length(arr_6))

  arr_3 = c(arr_3, rep(NA, max.len - length(arr_3)))
  dep_3= c(dep_3, rep(NA, max.len - length(dep_3)))
  arr_4= c(arr_4, rep(NA, max.len - length(arr_4)))
  dep_4= c(dep_4, rep(NA, max.len - length(dep_4)))

  df = data.frame(arr_1, dep_1,arr_2, dep_2, arr_3, dep_3, arr_4, dep_4, arr_
      5, dep_5, arr_6, dep_6)

  df_new = data.frame(times =
c(df[,"arr_1"],df[,"dep_1"],df[,"arr_2"], df[,"dep_2"],
df[,"arr_3"], df[,"dep_3"],df[,"arr_4"], df[,"dep_4"],
df[,"arr_5"], df[,"dep_5"],df[,"arr_6"], df[,"dep_6"]),
                            transactions =
c( rep("arr_1", length(arr_1)), rep("dep_1", length(dep_1)),
rep("arr_2", length(arr_2)), rep("dep_2", length(dep_2)),
rep("arr_3", length(arr_3)), rep("dep_3", length(dep_3)),
rep("arr_4", length(arr_4)), rep("dep_4", length(dep_4)),
rep("arr_5", length(arr_5)), rep("dep_5", length(dep_5)),
rep("arr_6", length(arr_6)), rep("dep_6", length(dep_6)) ))

  #sorting the data and removing the NA values
  sorted_df_new= data.frame(df_new[ with(df_new, order(df_new$times)),])
  sorted_df_new <- sorted_df_new[!(is.na(sorted_df_new$times)) ,]

  #finding the transactions
  T= c(0,0,0,0,0,0)
  df_queue_length = data.frame(
    times = rep(0, nrow(sorted_df_new)),
    transactions = rep(0, nrow(sorted_df_new)),
    q1 = rep(0, nrow(sorted_df_new)),
    q2 = rep(0, nrow(sorted_df_new)),
    q3 = rep(0, nrow(sorted_df_new)),
    q4 = rep(0, nrow(sorted_df_new)),
    q5 = rep(0, nrow(sorted_df_new)),
    q6 = rep(0, nrow(sorted_df_new))
    )

  df_queue_length$times = sorted_df_new$times
  df_queue_length$transactions = sorted_df_new$transactions

  t_list= list(rep(c(0),length(sorted_df_new[,1])))
  for(i in 1:length(sorted_df_new$times)){
    if(sorted_df_new$transactions[i]=="arr_1"){
      T[1]= T[1]+1
      t_list[[i]]=T
    }
    if(sorted_df_new$transactions[i]=="dep_1"){
```

```
    T[1]= T[1]-1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "arr_2"){
    T[2]= T[2]+1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "dep_2"){
    T[2]= T[2]-1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "arr_3"){
    T[3]= T[3]+1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "dep_3"){
    T[3]= T[3]-1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "arr_4"){
    T[4]= T[4]+1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "dep_4"){
    T[4]= T[4]-1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "arr_5"){
    T[5]= T[5]+1
    t_list[[i]]=T
  }
  if(sorted_df_new$transactions[i]== "dep_5"){
    T[5]= T[5]-1
```

```
      t_list[[i]]=T
    }
    if(sorted_df_new$transactions[i]== "arr_6"){
      T[6]= T[6]+1
      t_list[[i]]=T
    }
    if(sorted_df_new$transactions[i]== "dep_6"){
      T[6]= T[6]-1
      t_list[[i]]=T
    }
  }

  mtx = cbind(matrix(unlist(t_list), nrow= length(t_list),ncol= length(t_list
      [[1]]), byrow = TRUE))
  colnames(mtx) = c("length1", "length2","length3", "length4", "length5", "
     length6")

  for(i in seq(3,8)) {
    df_queue_length[,i] = mtx[,i-2]
  }

  df_queue_length$diff_time = c(diff(df_queue_length$times),100) == 0

  df_queue_length = df_queue_length %>% filter(!(df_queue_length$diff_time))
      %>% select(-diff_time, -transactions, -times)

  return(df_queue_length)
}
```

In order to shorten the computation time we converted one part of the code to C++ and used Rcpp package for calling it inside R. We converted it for the "for" and multiple "if" loops we have. You may see the code below.

```
    #include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadilloExtensions/sample.h>
#include <math.h>

using namespace Rcpp;

// [[Rcpp::export]]
arma::mat QueLen(NumericVector a) {
  int L = a.size();
  arma::mat len(L,6);
  len.zeros();
  for (int i = 0; i< L; i ++){
    int j= (a[i]+1)/2;
    int remain=a[i]+1-2*j;
    if(i!=0){
      for (int k = 0; k< 6; k ++){len(i,k)=len(i-1,k);}
      if(remain==0){
        len(i,j-1)=len(i-1,j-1)+1;
      } else {
        len(i,j-1)=len(i-1,j-1)-1;
      }
    }else{
      if(remain==0){len(0,j-1)=1;}
    }
  }
  return len;
}

// [[Rcpp::export]]
std::vector<int> find_locs(NumericVector a, double t){
  std::vector<int> id_locs;
  for(int i =0; i < a.size(); i++){
    if(a[i] == t){
      id_locs.push_back(i);
    }
  }
  return id_locs;
}

// [[Rcpp::export]]
double Phi(double x, double y){
  double u = std::log(x) - std::log(y);
  return u;
}

// [[Rcpp::export]]
Rcpp::NumericVector Count_C(arma::mat x, int L){
  Rcpp::NumericVector aa = rep(0.,6);
  Rcpp::NumericVector bb = rep(0.,8);
  for (int i = 1; i< L; i ++){
    for (int j = 0; j< 6; j ++){aa[j]=x(i,j)-x(i-1,j);}
    double sum=std::accumulate(aa.begin(), aa.end(), 0.0);
    if(sum==0){
      std::vector<int> index1 = find_locs(aa,-1);
      std::vector<int> index2 = find_locs(aa,1);
      if (index2[0]==4){
        if(index1[0]==2){bb[4]++;} else {bb[5]++;}
      } else if (index2[0]==5){
        bb[6]++;
      } else {
        bb[index2[0]]++;
      }
    } else if (sum==1){
      bb[0]++;
    } else {
      bb[7]++;
    }
  }

  return bb;
}

// [[Rcpp::export]]
Rcpp::NumericVector CUSUM(arma::mat x, int L){
  Rcpp::NumericVector aa = rep(0., L);
  for (int i = 1; i< L; i ++){
    arma::mat xx=x.rows(0,i);
    //std::cout << "length= " << xx.n_rows << '\n';
    Rcpp::NumericVector C= Count_C(xx, i+1);
    double l1=Phi(1,2.1)*C[0]+Phi(1.1,2.1)*C[1]+Phi(0.6*1.1,2.2)*C[2]+Phi(0.4*
        1.1,2.2)*C[3]+Phi(1.1,1.1+0.6*1.1)*C[4]+Phi(1.1,1.1+0.4*1.1)*C[5]+Phi
        (1.1,3.3)*C[6]+Phi(1.1,2.2)*C[7];
    double l0=Phi(1,1.99)*C[0]+Phi(0.99,1.99)*C[1]+Phi(0.6*0.99,0.99+0.99)*C
        [2]+Phi(0.4*0.99,0.99+0.99)*C[3]+Phi(0.99,0.99+0.6*0.99)*C[4]+Phi
        (0.99,0.99+0.4*0.99)*C[5]+Phi(0.99,0.99+0.99+0.99)*C[6]+Phi
        (0.99,0.99+0.99)*C[7];
    double gg= l1-l0;
    double g= max(NumericVector::create(0.,gg));
    aa[i]=aa[i-1]+g;
  }
  return aa;
}
```

A.1.2 Simulation for Computing Likelihood Ratio

You may see the long form of the code that we have used while computing likelihood ratios below.

```
    df = QL_simulator(lambdas, mus, t_lim)
Likelihood_ratio = function(df, lambdas, mus) {
  c_1 = 0
  c_12 = 0
  c_23 = 0
  c_24 = 0
  c_35 = 0
  c_45 = 0
  c_56 = 0
  c_6 = 0

  for(i in seq(nrow(df) -1)) {
    if(prod(df[i+1, ] - df[i, ] == c(1,0,0,0,0,0)) == 1) c_1 = c_1 + 1
    if(prod(df[i+1, ] - df[i, ] == c(-1,1,0,0,0,0)) == 1) c_12 = c_12 + 1
    if(prod(df[i+1, ] - df[i, ] == c(0,-1,1,0,0,0)) == 1) c_23 = c_23 + 1
    if(prod(df[i+1, ] - df[i, ] == c(0,-1,0,1,0,0)) == 1) c_24 = c_24 + 1
    if(prod(df[i+1, ] - df[i, ] == c(0,0,-1,0,1,0)) == 1) c_35 = c_35 + 1
    if(prod(df[i+1, ] - df[i, ] == c(0,0,0,-1,1,0)) == 1) c_45 = c_45 + 1
    if(prod(df[i+1, ] - df[i, ] == c(0,0,0,0,-1,1)) == 1) c_56 = c_56 + 1
    if(prod(df[i+1, ] - df[i, ] == c(0,0,0,0,0,-1)) == 1) c_6 = c_6 + 1
  }

  r_1 = log(lambdas[1]) - log(lambdas[1] + mus[1])
  r_12 = log(mus[1]) - log(lambdas[1] + mus[1])
  r_23 = log(0.6*mus[2]) - log(0.6*mus[2] + 0.4*mus[2])
  r_24 = log(0.4*mus[2]) - log(0.6*mus[2] + 0.4*mus[2])
  r_35 = log(mus[3]) - log(mus[3] + 0.6*mus[2])
  r_45 = log(mus[4]) - log(mus[4] + 0.4*mus[2])
  r_56 = log(mus[5]) - log(mus[4] + mus[5])
  r_6 = log(mus[6]) - log(mus[5] + mus[6])

  l0 =
 (r_1*c_1+r_12*c_12+r_23*c_23+r_24*c_24+r_35*c_35+r_45*c_45+r_56*c_56+r_6*c_6)

  mus = 0.9*c(1.1,1.1,1.1,1.1,1.1,1.1)
  r_1 = log(lambdas[1]) - log(lambdas[1] + mus[1])
  r_12 = log(mus[1]) - log(lambdas[1] + mus[1])
  r_23 = log(0.6*mus[2]) - log(0.6*mus[2] + 0.4*mus[2])
  r_24 = log(0.4*mus[2]) - log(0.6*mus[2] + 0.4*mus[2])
  r_35 = log(mus[3]) - log(mus[3] + 0.6*mus[2])
  r_45 = log(mus[4]) - log(mus[4] + 0.4*mus[2])
  r_56 = log(mus[5]) - log(mus[4] + mus[5])
  r_6 = log(mus[6]) - log(mus[5] + mus[6])

  l1 =
 (r_1*c_1+r_12*c_12+r_23*c_23+r_24*c_24+r_35*c_35+r_45*c_45+r_56*c_56+r_6*c_6)

  return(l1 - l0)
}
```

Computing time of this code was slow, that's why we eliminated the loops and wrote a new code. You may see the faster version below.

```
    likelihood_ratio_cal = function(df, lambdas,mus) {
  r_1 = log(lambdas[1]) - log(lambdas[1] + mus[1])
  r_12 = log(mus[1]) - log(lambdas[1] + mus[1])
  r_23 = log(0.6*mus[2]) - log(0.6*mus[2] + 0.4*mus[2])
  r_24 = log(0.4*mus[2]) - log(0.6*mus[2] + 0.4*mus[2])
  r_35 = log(mus[3]) - log(mus[3] + 0.6*mus[2])
  r_45 = log(mus[4]) - log(mus[4] + 0.4*mus[2])
```

```
r_56 = log(mus[5]) - log(mus[4] + mus[5])
r_6 = log(mus[6]) - log(mus[5] + mus[6])


mus_new = 0.9*mus
rr_1 = log(lambdas[1]) - log(lambdas[1] + mus_new[1])
rr_12 = log(mus_new[1]) - log(lambdas[1] + mus_new[1])
rr_23 = log(0.6*mus_new[2]) - log(0.6*mus_new[2] + 0.4*mus_new[2])
rr_24 = log(0.4*mus_new[2]) - log(0.6*mus_new[2] + 0.4*mus_new[2])
rr_35 = log(mus_new[3]) - log(mus_new[3] + 0.6*mus_new[2])
rr_45 = log(mus_new[4]) - log(mus_new[4] + 0.4*mus_new[2])
rr_56 = log(mus_new[5]) - log(mus_new[4] + mus_new[5])
rr_6 = log(mus_new[6]) - log(mus_new[5] + mus_new[6])


df1 = df[-1,]
df0 = df[-nrow(df),]
diff_df = df1 - df0
diff_df = diff_df %>%
mutate(c1 = (q1 ==1 & q2 == 0 & q3 ==0 & q4 == 0 & q5 == 0 & q6 == 0)) %>%
mutate(c12 = (q1 == -1 & q2 == 1 & q3 ==0 & q4 == 0 & q5 == 0 & q6 == 0))
   %>%
mutate(c23 = (q1 == 0 & q2 == -1 & q3 == 1 & q4 == 0 & q5 == 0 & q6 == 0))
   %>%
mutate(c24 = (q1 == 0 & q2 == -1 & q3 == 0 & q4 == 1 & q5 == 0 & q6 == 0))
   %>%
mutate(c35 = (q1 == 0 & q2 == 0 & q3 == -1 & q4 == 0 & q5 == 1 & q6 == 0))
   %>%
mutate(c45 = (q1 == 0 & q2 == 0 & q3 == 0 & q4 == -1 & q5 == 1 & q6 == 0))
   %>%
mutate(c56 = (q1 == 0 & q2 == 0 & q3 == 0 & q4 == 0 & q5 == -1 & q6 == 1))
   %>%
mutate(c6 = (q1 == 0 & q2 == 0 & q3 == 0 & q4 == 0 & q5 == 0 & q6 == -1))
   %>%
select(c1,c12,c23,c24,c35,c45,c56,c6) %>%
  mutate(c1 = cumsum(c1), c12 = cumsum(c12), c23 = cumsum(c23), c24 = cumsum(
      c24), c35 = cumsum(c35), c45 = cumsum(c45), c56 = cumsum(c56), c6 =
      cumsum(c6)) %>%
  mutate(l10 =
 (rr_1*c1+rr_12*c12+rr_23*c23+rr_24*c24+rr_35*c35+rr_45*c45+rr_56*c56+rr_6*c6)
      - (r_1*c1+r_12*c12+r_23*c23+r_24*c24+r_35*c35+r_45*c45+r_56*c56+r_6*c6))
  return(diff_df$l10)
}
```

A.1.3 Simulation for Obtaining Average Run Length

As a result of our simulation we seek to obtain the average run lengths at the threshold value from the CUMSUM charts. You may see the full code below to obtain ARL.

```
RL = function(h) {
  df = QL_simulator(lambdas, mus, t_lim)
  x = likelihood_ratio_cal(df, lambdas, mus)
  y = cusum_auxiallry(x)
  z = which(y > h)
  z[1]
}
 h= c(0.3178,0.3182,0.3184,0.3186) # this may vary
 ARL_0 =c()
 RLs_list= list()

for(i in 1:(length(h))){
  RLs_list[[i]]  = unlist(replicate(10000,RL(h[i])))
  ARL_0[i] = mean(RLs_list[[i]])
}
K = cbind(h, ARL_0)
```