VRIJE UNIVERSITEIT AMSTERDAM
Faculty of Science - Mathematics Department
Master in Pure Mathematics - Probability and Statistics

Fractional staffing levels in Erlang models

Daily supervisor: Alex Roubos
First Supervisor: Gabriele Benedetti
Second Reader: Rene Bekker

Thesis of:
Elena Garlati
Student Number 2691820

CONFIDENTIAL REPORT

March 27, 2023

Management Summary

The field of Workforce Management entails to improve and optimize the capacity planning for well-functioning workforce systems, such as contact centers. In the capacity management part of the WFM process, safety staffing decisions are handled by using the Erlang models: advanced mathematical models used for studying queueing systems. This mathematical theory consists of stochastic processes, such as Markov chains, that enable to study the performance measures of queueing systems. However, these stochastic processes allow to only work with integer staffing levels. This is a problem in reality because with a non-work-conserving policy, it is not guaranteed that all the scheduled agents of a certain shift are always present in the system. Due to this shrinkage phenomenon, the number of agents needs to be increased in order to guarantee a good performance of the system. Subsequently, additional rounding needs to be made in the scheduling part of the WFM process. Rounding twice the staffing level leads to an increase in staffing costs. For this reason, the project aims to identify new methods for the computation of fractional staffing levels in Erlang models.

With this research, we give an overview on which are the different approaches to extend the Erlang models to a continuous setting in terms of staffing levels. The model on which we mainly focused for our research is Erlang C. First, we derive the re-formulation of the performance measures as continuous functions in terms of the number of agents from a theoretical point of view. This is not applicable to all the Erlang models. Therefore, in the project's second part, we introduce new approaches for the research of fractional staffing levels in Erlang C models to extend this problem to other more complicated systems. With this research, we eventually prove that it is possible to obtain fractional staffing levels for Erlang systems that exactly meet certain targets.

Contents

Chapter 1

Introduction

This research has the goal to answer important questions on how stochastic models describe certain queueing systems, such as call centers. In particular, we will discuss new approaches for extending the traditional expression of Erlang models for the optimization of call center workforce planning.

1.1 Business context of the problem

This project is commissioned by CCmath, a software development company that was founded in 2005 with the goal to make Call Center Mathematics more accessible. Some of the offered services are forecasting and workforce planning software, (online) training, and advice, for an expanding international customer base. The main goal behind it is to provide software and tools so that every customer company can improve significantly the efficiency level of the Workforce Management (WFM) process by taking advantage of advanced mathematical methods. The complicated WFM mathematics was reduced to more accessible tools such as CCforecast, Erlang X, and Erlang Chat. These tools enable a customer to reduce the number of hours spent forecasting using advanced AI in order to save more time for gathering more relevant information and sharing analyses with stakeholders. Indeed, the idea is to multiply the productivity and effectiveness of any WFM team by obtaining lower costs while improving the service level and employee satisfaction.

In call centers, service quality objectives determine the staffing level. For instance, the managers of a call center might want to guarantee a service level of 80% within a certain amount of time. In particular, they want the 80% of the customers to be served in less than 20 seconds or with an average speed of answer of 1 minute. However, in practice, the staffing level is an integer number and therefore in most cases, it is not possible to select a staffing level such that the

objective is perfectly achieved. The various types of software that are available at CCmath manage to make predictions on customer demand, operate on capacity management, and handle the scheduling process.

1.1.1 Workforce Management

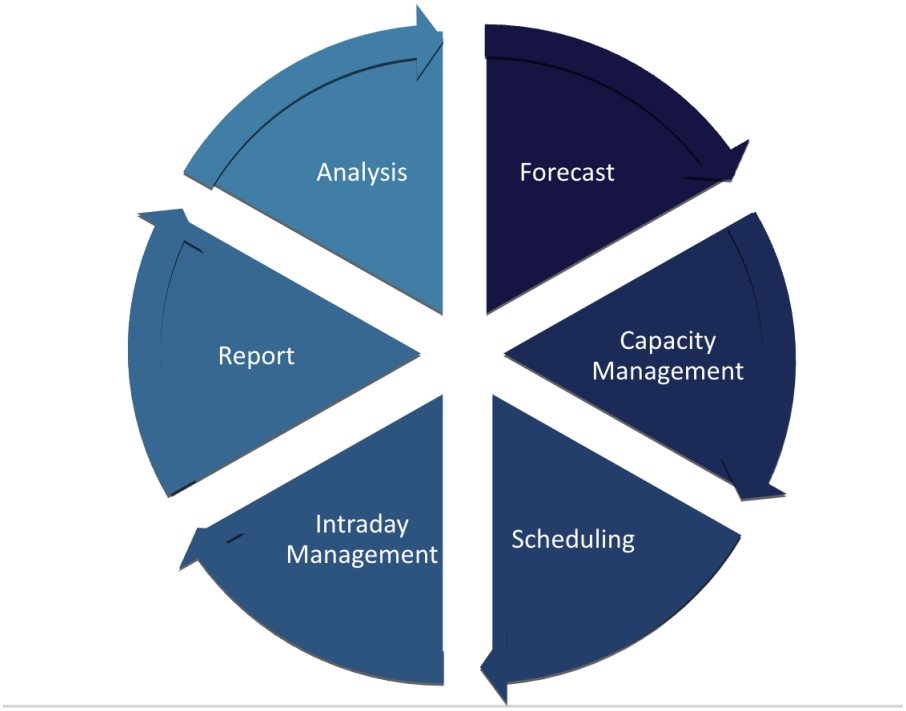

Figure 1.1.1: WFM cycle.

The business field of this project is Workforce Management, which by advanced mathematics aims to describe all the steps required to make efficient workforce planning. These steps are defined in the WFM cycle presented in Figure 1.1.1 with the following order: forecasting, capacity management, scheduling, intra-day management, report, and analysis. By following this cycle, it is possible to improve workforce planning in order to guarantee that the capacity of a certain system is as balanced as possible. In the first step of the cycle, the goal is to make predictions on certain parameters of the system, such as the volume of calls or the average handling time. By using the output of the forecasting step, it is possible to manage the capacity of a certain system with staffing decisions. Next, the scheduling step of the cycle is where the shifts are actually made by considering shrinkage factors and by keeping a balance between the customers, the employees, and the business of the system. Subsequently, the goal of the intra-day management part is to

understand how to deal with the randomness during the day achieving the average service level. Finally, with the report and the analysis steps it is possible to collect results and analyze the performance of the previous steps in order to optimize the process and make a new forecast to start the cycle again.

This project focuses on the capacity management part, which is essential for the following step of scheduling, where the shifts of agents are determined according to different factors. The main goal of the capacity management part is to ensure optimized levels of staff against business demands at any given time. This step of the process is also called *safety staffing* since we focus on planning overcapacity in order to deal with fluctuations in workload. This part concentrates on the realization of the gross workforce, which is defined as the sum of the shrinkage and the gross workload. In particular, it is possible to compute the gross workload by adding the safety Erlang calculation to the net workload, i.e., the fraction between the volume of the forecast and the service rate of a single server.

Both in the capacity management and in the scheduling steps some roundings of the number of agents are made and this leads to an increase in staffing costs. Indeed, the number of agents is rounded twice, once in each step. In order to avoid any extra approximation, the aim of this research is to find different methods to give as an output of the capacity management the real value of the staffing level that exactly achieves the service level objective. In this way, it will be possible to round this number only in the scheduling part and therefore avoid any extra approximation.

1.2 Research problem

In call centers, classic queuing models are used in order to compute various performance measures, such as the service level or the average waiting time, where the staffing level is decided with service quality objectives. For instance, the call center goal might be to obtain 80% of served calls within 20 seconds or with an average speed of answer of 1 minute. Staffing decisions always rely on queuing models such as Erlang A or C and the staffing level that is output by the Erlang models is an integer number. This is a problem because the agents are working according to a non-work-conserving policy in reality. For instance, consider regular shifts in a call center and assume that in a certain shift, there are s agents working at the same time. With a non-work-conserving policy, the presence of all agents for all the shifts is not guaranteed, because someone might need to take a short break for example. The phenomenon that not all scheduled agents are either working on a call or are available to take a call is called shrinkage, which can be as large as 30%. By taking the shrinkage into account, we transform the net requirement (i.e., the output of the Erlang models) into a gross requirement.

For scheduling purposes, the gross requirement needs to be an integer. However, when the net requirement is also limited to an integer, we are effectively rounding twice and thereby unnecessarily increasing the staffing costs. In the end, the goal is to find the net requirement expressed in a fractional number of agents such that the service quality objectives are perfectly achieved.

1.3 Scientific background

As already mentioned, staffing decisions always rely on queueing models such as Erlang A or C, where the number of agents that work in the system are usually homogeneous, identical, and independent and they serve customers with the first-come-first-served discipline. This hypothesis is assumed to be true in order to avoid the complexity of the arrival process, service time and patience distributions, or other non-stationary behaviors. Therefore, Markov chains are used to define these models, where their probability distribution $\pi(x)$ describes the probability of having exactly x customers in the system.

The capacity management software at CCmath uses some of these models and other slight variations of them. In particular, Erlang C is the first model used, where customers who cannot find an available agent wait in an infinite-capacity queue. There is also a slight variation of this model, which is called Erlang CL (Lines) and it considers a finite queue, meaning that the maximum number of customers allowed in the system is a finite number N. Another model is Erlang X, which is an extension of Erlang CL: in addition to a finite queue, we also consider abandonment. Therefore, we regard the patience of customers to be exponentially distributed with parameter γ, and from all the customers that abandon the system, we see that a fraction α of them enter the system again. This determines the phenomenon of retrials. There are also other versions of Erlang X, however, they will not be the topic of this research.

This project will require also some knowledge of analysis: in particular, the first part of the project shows proof of the equivalence between the discrete and the continuous formulas of the performance measures in terms of the staffing level. In order to be able to implement integrals via numerical integration techniques instead of using the Markov chains formulas, further knowledge of numerical analysis is required. We need to make this approximation because C++, the current programming language used by the company, does not allow us to compute continuous values as integrals. In this implementation part, another branch of numerical analysis will be essential: convergence methods. For these new approaches, an analysis of accuracy has been conducted by using the exact real values for staffing level derived from integrals calculated with other programming languages that allow computing them without any approximation, such as Python.

Subsequently, we want to find new approaches to approximate the real number of agents when we have integer values and we want to find appropriate interpolating functions. This last approach will be the bridge between the different models: from Erlang C to Erlang CL and Erlang X.

1.4 Literature review

The probabilistic field presents several studies and researches about the Erlang models and more in general about Markov Chains. For this project, we referred to different sources since in addition to the theoretical area of Erlang models, other technical and mathematical tools are used, for instance, numerical integration methods, convergence methods, and interpolation functions.

Regarding the probabilistic and business background in contact centers the main source used is the book titled "Call Center Optimization" by Koole [1] which discusses methods for optimizing call center performance. It provides an overview of the key challenges and factors that impact call center performance and describes a range of techniques for addressing these challenges and optimizing call center operations. In particular, it gives important insights regarding the queueing systems described by Erlang C and Erlang X models. It is also one of the main sources for the WFM background studied for this project, by discussing the importance of call centers as a key interface between companies and their customers, and the impact of call center performance on customer satisfaction and business outcomes. Important related references to this topic are "A practice-oriented overview of call center workforce planning" by Koole and Li [2] and "Telephone call centers: Tutorial, review, and research prospects" by Gans, Koole, and Mandelbaum [3].

From the theoretical point of view, another important source is "Markov chain models of a telephone call center with call blending" by Leemis and Park [4], referring to the process of assigning incoming calls to agents who are trained to handle multiple types of calls. This article provides an overview of call centers and call blending, and describes the benefits and challenges of call blending, including improved efficiency and flexibility, but also increased complexity in modeling and analysis. The authors use the Markov chain model to analyze the performance of the call center under different scenarios, including varying levels of call arrival rates, agent staffing levels, and call routing strategies. The authors also use the model to evaluate the impact of different performance metrics, such as average waiting time, on the overall performance of the call center. This article is mainly used in this research for studying the theory behind the Erlang models as presented later in Chapter 2.

Subsequently, regarding the implementation part of this project, the main sources used to provide complex mathematical tools for numerical analysis, for

instance "Introduction to Numerical Analysis" by Levy [5]. In this book, the central topics for this project are presented as numerical integration methods or interpolation approaches. Another important reference for numerical analysis theory is "Iterative methods for linear and nonlinear equations" by Kelley [6], where a key topic for this research is presented: the convergence methods. Indeed, this book provides knowledge on iterative methods later presented in Chapter 4 of this thesis.

The goal of this research is not a widely discussed topic in literature. Therefore, the material previously mentioned and other additional papers gave us the basic methodologies and knowledge to undertake the research of fractional values for staffing levels in Erlang systems.

1.5 Structure of the thesis

After this introduction to the problem of the research, the project will be presented in more detail explaining and analyzing every step of this research.

In Chapter 2, there is a further and more detailed introduction to the Erlang models used for this project. In particular, there will be a theoretical description of these systems when described by the Markov Chains, therefore in a discrete setting in terms of the staffing level.

Chapter 3 will concentrate on the continuous formulas for Erlang C and Erlang CL models with related proofs of equality with the discrete stationary distribution and performance measures.

Next, in Chapter 4 there will be an overview of the implementation approaches for the continuous formulas of the Erlang C model. In particular, when the number of agents is given and we are willing to compute the value of the performance measures, we introduce the numerical integration methods. On the other hand, when a certain target for a performance measure is given and we want to compute the corresponding fractional staffing level in order to achieve that specific target, we use the convergence measures.

In Chapter 5, we will present an additional approach for the research of the number of agents in Erlang C models. The interpolation approach consists of the analysis of both linear and quadratic functions and we apply this approach for the two different scenarios: computing the performance measure for a given staffing level and finding the fractional number of agents for a given target of one of the performances.

Finally, in Chapter 6 we will discuss the conclusion of the project by presenting the limitations of this research and the possible future approaches to this problem. In particular, we will discuss how it is possible to extend the interpolation approach to Erlang CL and Erlang X models based on the results achieved in this research.

Chapter 2

The Erlang systems

The dynamics of a single-skilled inbound call center are crucial elements for studying the performance measures of the systems analyzed in this project. Specific characteristics will be considered for each model, starting from a more idealistic situation to continuing with more and more realistic scenarios. Queuing theory is essential for describing their complexity and for making predictions about their performance behavior. In particular, the Erlang formulas represent most of the theory applied in these scenarios, therefore we can call them Erlang systems.

A combination of technology and human behaviors plays an important role in complex systems such as call centers. The multiplicity of features of these difficult systems cannot always be represented by taking all these different aspects into account. Therefore, in order to make a prediction of the performance of such systems we need to consider a simplification of reality, which can be made by using mathematical models.

In this project, the Erlang systems that will be studied are the following: *Erlang C*, *Erlang CL*, and *Erlang X* models.

2.1 The Erlang C model

The first crucial model that we need to consider for describing complex systems such as call centers is certainly the Erlang C model. Stated by the Danish mathematician A. Erlang in 1917, this model is a capacity planning method that allows dealing with workforce management.

In order to have a relevant model, it has to describe all the essential features that build the system we want to study. Regarding the Erlang C model, the most important characteristic is that both call arrival times and handling times are random quantities. Moreover, the features that we have to consider are the following:

- arriving calls come independently according to a Poisson process with parameter λ;
- service duration follows an Exponential distribution with parameter μ;
- the system provides a fixed number of agents s for a certain time interval, where these agents are identical and independent;
- when all servers are busy, incoming calls wait in an infinite-length queue until they get served, which means that customers cannot abandon the system before the agents serve them;
- incoming calls are served with FCFS (first come first served) policy, therefore the longest-waiting call gets served first.

2.1.1 Stationary distribution

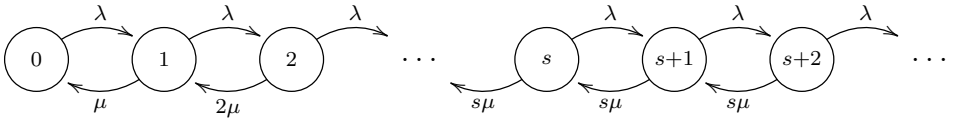

Figure 2.1.1: Markov chain of Erlang C model, where λ is the arrival rate and μ is the service rate. The system has infinite capacity.

In order to describe the Erlang C model, it is necessary to represent it as a birth-death process, therefore as a Markov chain. As already mentioned, queuing theory is essential to analyze this model: the state space and the transition diagram of this Markov chain are represented in Figure 2.1.1. The state space of this process is $S = \{0, 1, \dots\}$ and represents the number of customers in the system. It is possible to "jump" from one state to the next ones with different rates. Assume there are s servers in the system, therefore if there are $x < s$ customers, they will be served immediately and the rate by which it is possible to go down in the Markov chain is $x \cdot \mu$ because there are x agents working at the same time. On the other hand, if the number of customers x is greater than the number of agents s we certainly have that the rate of going down is $s \cdot \mu$ for all $x \geq s$ because the maximum number of servers handling calls at the same time is s. Therefore the Markov chain presented in Figure 2.1.1 is an $M|M|s$-queue, where the first M represents the interarrival process with exponential distribution of parameter λ, the second M represents the exponential distribution with parameter μ for the handling time and s is the number of agents working in the system.

Going into more detail, we can explicitly compute the stationary distribution $\pi(x)$ for all $x \in S$, i.e., the probability of having x customers in the system. We can compute it by using the *Balance Principle*, which states that the number of transitions out of a state should be equal to the number of transitions into the same state. Notice first that we can define the load of the system as $a = \frac{\lambda}{\mu}$. Therefore, we have:

- if $x < s$:

$$\begin{aligned} \lambda\pi(x) &= \mu(x+1)\pi(x+1) \\ \pi(x+1) &= \frac{\lambda}{\mu}\frac{1}{x+1}\pi(x) \\ \implies \pi(x) &= \frac{a^x}{x!}\pi(0); \end{aligned}$$

- if $x \geq s$:

$$\begin{aligned} \lambda\pi(x) &= \mu s\pi(x+1) \\ \pi(x) &= \left(\frac{a}{s}\right)^{x-s}\pi(s) \\ \pi(x) &= \left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\pi(0), \end{aligned}$$

where we know from the first case that $\pi(s) = \frac{a^s}{s!}\pi(0)$. We also know that $\sum_{x=0}^{\infty}\pi(x) = 1$, because $\pi(x)$ is the probability of having x customers in the system, therefore over the whole state space S these probabilities sum up to 1.

Then by normalization, we obtain the following value for $\pi(0)$:

$$\begin{aligned}
\sum_{x=0}^{\infty}\pi(x) &= \sum_{x=0}^{s-1}\pi(x) + \sum_{x=s}^{\infty}\pi(x) \\
&= \sum_{x=0}^{s-1}\frac{a^x}{x!}\pi(0) + \sum_{x=s}^{\infty}\left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\pi(0) \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\sum_{y=0}^{\infty}\left(\frac{a}{s}\right)^{y}\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\frac{1}{1-\frac{a}{s}}\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\frac{s}{s-a}\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{(s-1)!(s-a)}\right] \\
\implies \pi(0) &= \left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1}.
\end{aligned}$$

Notice that the geometric series $\sum_{y=0}^{\infty}\left(\frac{a}{s}\right)^y$ converges if and only if $\frac{a}{s} < 1$, therefore we need to guarantee that $s > a$. We will see in the discussion about performance measures that this assumption is essential also for the stability of the system.

Therefore, the stationary distribution of the Erlang C model is presented in the following proposition.

Proposition 2.1.1. *Given a state space $S = \{0, 1, 2, \dots\}$ and a value $s \in \mathbb{Z}_+$ such that $s > a = \frac{\lambda}{\mu}$, the Erlang C model defined on this state space has stationary distribution π defined as follows:*

- *if $x < s$:*
$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1};$$
- *if $x \geq s$:*
$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1};$$

for all $x \in S$.

2.1.2 Performance measures

In the Erlang C model, we are interested in some performance measures, such as the service level, the average waiting time and the occupancy of the system.

Service Level

Definition 2.1.1 (Service level)**.** *Given W the waiting time of a single customer in an Erlang C model and τ the acceptable waiting time, the service level (SL) is defined by the following probability*

$$\mathrm{SL} = \mathbb{P}(W \leq \tau).$$

The service level is defined by the percentage of a group of calls that are answered within a certain amount of time τ that can be considered as a threshold. Because of the randomness of the system, the value of the service level might change if we consider different groups of customers. However, by computing the average over infinitely many calls we always obtain the same number. This is possible via the Erlang C formula, which takes as input the number of agents in the system, the arrival rate, and the service rate. The relevant quantity in this formula is the load $a = \frac{\lambda}{\mu}$, which indeed is the fraction between the arrival rate and the service rate. In particular, the load of the system represents the number of agents that are needed to manage all traffic. A number s of agents is selected to deal with the offered traffic. We need to make an important assumption that guarantees that the value s is greater than the load a, otherwise, we would run into instability. On average the number of arrivals should be less than departures per time unit. For those values of s that lead to an unstable system the service level is considered to be 0%. Therefore, the difference between s and a defines a new quantity called the overcapacity of the system. This quantity needs to control the variations of the offered load, which are not caused by the rates of arrivals or departures, but come from the random behavior of interarrival and call handling times themselves.

Furthermore, the performance of service is given by the Erlang C formula when the parameters λ and μ and the value s are given. In addition to these values, we need to consider also the acceptable waiting time τ that represents the threshold of time by which a single call should be answered.

Definition 2.1.2 (Probability of delay)**.** *Given the load $a = \frac{\lambda}{\mu}$ and the number of agents $s \in \mathbb{Z}_+$, the probability of delay in an Erlang C model is*

$$C(s, a) = \mathbb{P}(W > 0),$$

i.e., the probability that all servers are occupied.

Proposition 2.1.2. *Given the load $a = \frac{\lambda}{\mu}$ and the number of agents $s \in \mathbb{Z}_+$, for $a < s$ the service level can be written as*

$$\mathrm{SL} = 1 - C(s,a)e^{-(s\mu-\lambda)\tau}$$

where $C(s,a) = \mathbb{P}(W > 0)$ is the probability of delay.

Proof. Notice first that the exponential term comes from the definition of service level, which is indeed the probability that the waiting time is less than τ. In particular, the random variable $W|W > 0$ follows an Exponential distribution with parameter $s\mu - \lambda$. Therefore we have the following:

$$\begin{aligned}\mathbb{P}(W \leq \tau) &= 1 - \mathbb{P}(W > \tau) \\ &= 1 - \big(\mathbb{P}(W > \tau | W > 0)\mathbb{P}(W > 0) + \mathbb{P}(W > \tau | W = 0)\mathbb{P}(W = 0)\big) \\ &= 1 - C(s,a)e^{-(s\mu-\lambda)\tau}\end{aligned}$$

where of course $\mathbb{P}(W > \tau | W = 0) = 0$ and by definition $C(s,a) = \mathbb{P}(W > 0)$. □

This expression of the service level can be obtained by using the following proposition.

Proposition 2.1.3. *In an Erlang C system with s agents and load a such that $a < s$, the probability of delay can be computed as follows:*

$$C(s,a) = \frac{a^s}{(s-1)!(s-a)}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1}.$$

Proof. We can prove that from the definition of probability of delay, we can obtain the formula given in Proposition 2.1.3. By definition, we know that $C(s,a) = \mathbb{P}(W > 0)$, therefore we can compute $C(s,a)$ as the probability of finding all the agents occupied in the system, i.e.,

$$\begin{aligned}C(s,a) &= \sum_{x=s}^{+\infty}\pi(x) \\ &= \sum_{x=s}^{\infty}\left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1}\end{aligned}$$

where we explicitly substitute the results found in Proposition 2.1.1 for $x \geq s$. We

can continue the computation as follows:

$$\begin{aligned}
C(s,a) &= \sum_{x=s}^{\infty}\left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1} \\
&= \frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1}\sum_{x=s}^{\infty}\left(\frac{a}{s}\right)^{x-s} \\
&= \frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1}\sum_{i=0}^{\infty}\left(\frac{a}{s}\right)^{i} \\
&= \frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1}\frac{1}{1-\frac{a}{s}}
\end{aligned}$$

where since by assumption we have that $a < s$, the geometric series converges to the term $\frac{1}{1-\frac{a}{s}}$. The computation continues as follows:

$$\begin{aligned}
C(s,a) &= \frac{1}{1-\frac{a}{s}}\cdot\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1} \\
&= \frac{s}{s-a}\cdot\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1} \\
&= \frac{a^s}{(a-s)(s-1)!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!}+\frac{a^s}{(s-1)!(s-a)}\right]^{-1}
\end{aligned}$$

which is what we wanted to obtain. □

Average waiting Time

Another important performance measure used to study the Erlang C model behavior is the average speed of answer (ASA), i.e., the average amount of time that calls spend waiting.

Proposition 2.1.4. *Given the load $a = \frac{\lambda}{\mu}$ and the number of agents $s \in \mathbb{Z}_+$, for $a < s$ the average speed of answer can be written as*

$$\text{ASA} = \frac{C(s,a)}{s\mu-\lambda} = \frac{C(s,a)\cdot\frac{1}{\mu}}{s-a}$$

.

Notice that by this proposition the average waiting time is defined as the probability of delay, times the average handling time, over the overcapacity.

Proof. This formula derives again from the fact that $W|W > 0$ follows an exponential distribution with rate $s\mu - \lambda$. In particular,

$$\begin{aligned} \text{ASA} &= \mathbb{E}(W) \\ &= \mathbb{E}(W|W > 0)\mathbb{P}(W > 0) + \mathbb{E}(W|W = 0)\mathbb{P}(W = 0) \\ &= \frac{1}{s\mu - \lambda} \cdot C(s, a). \end{aligned}$$

□

Occupancy

The last performance measure of our interest for the Erlang C model is the occupancy of a single server, i.e., the rational number

$$\text{OCC} = \frac{\lambda}{s\mu} = \frac{a}{s}.$$

2.2 The Erlang CL model

The second system we are considering in this project is the Erlang CL model, which has many similarities with the previous one. However, it presents some new features that might also lead to new limitations. Erlang CL stands for Erlang C Lines, where "Lines" indicates that the queue has a finite capacity in this model. Therefore, we can consider a number N as the maximum number of customers that are allowed to be in the system at the same time. In this system, any customer can be rejected if the system is full, i.e., if s customers are being served and $N - s$ customers are already waiting in the queue.

2.2.1 Stationary distribution

In Figure 2.2.1, the transition diagram of the Erlang CL Markov chain is shown. As already mentioned, the queue has finite capacity and only N customers are accepted in the system at the same time, therefore the $(N + 1)$-th customer is rejected with blocking probability $\pi(N)$. Notice that the state space of this process is not infinite anymore, it has a finite number of elements: $S = \{0, 1, \ldots, N\}$. As we showed for Erlang C, we can compute the stationary distribution of this Markov chain by using the *Balance Equation* as follows:

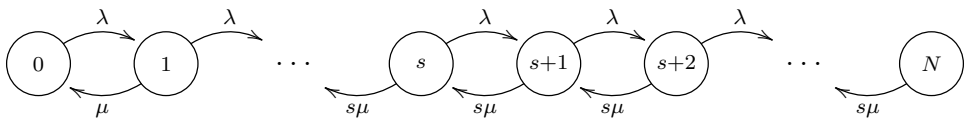

Figure 2.2.1: Markov chain of Erlang CL model where λ is the arrival rate and μ is the service rate. The maximum capacity of the system is N.

- if $x < s$:

$$\begin{aligned} \lambda\pi(x) &= \mu(x+1)\pi(x+1) \\ \pi(x+1) &= \frac{\lambda}{\mu}\frac{1}{x+1}\pi(x) \\ \implies \pi(x) &= \frac{a^x}{x!}\pi(0); \end{aligned}$$

- if $s \leq x \leq N$:

$$\begin{aligned} \lambda\pi(x) &= \mu s\pi(x+1) \\ \pi(x) &= \left(\frac{a}{s}\right)^{x-s}\pi(s) \\ \pi(x) &= \left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\pi(0). \end{aligned}$$

We know that the probabilities sum up to 1, i.e., $\sum_{x=0}^{N} \pi(x) = 1$, then by normalization we obtain the following:

$$\begin{aligned}
\sum_{x=0}^{N} \pi(x) &= \sum_{x=0}^{s-1} \pi(x) + \sum_{x=s}^{N} \pi(x) \\
&= \sum_{x=0}^{s-1} \frac{a^x}{x!}\pi(0) + \sum_{x=s}^{N} \left(\frac{a}{s}\right)^{x-s} \frac{a^s}{s!}\pi(0) \\
&= \pi(0)\left[\sum_{x=0}^{s-1} \frac{a^x}{x!} + \frac{a^s}{s!}\left(\frac{a}{s}\right)^{-s} \sum_{x=s}^{N} \left(\frac{a}{s}\right)^{x}\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1} \frac{a^x}{x!} + \frac{a^s}{s!}\left(\frac{a}{s}\right)^{-s} \frac{\left(\frac{a}{s}\right)^{N+1} - \left(\frac{a}{s}\right)^{s}}{\frac{a}{s} - 1}\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1} \frac{a^x}{x!} + \frac{a^s}{s!} \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right] \\
\implies \pi(0) &= \left[\sum_{x=0}^{s-1} \frac{a^x}{x!} + \frac{a^s}{s!} \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right]^{-1}
\end{aligned}$$

where we use the equality $\sum_{j=m}^{n} x^j = \frac{x^{n+1} - x^m}{x-1}$ with $x \neq 1$. If $x = 1$, i.e., if $a = s$, we obtain different results for the stationary distribution. This particular case is explained in Appendix A.

Therefore, the stationary distribution of the Erlang CL model is presented in the following proposition.

Proposition 2.2.1. *Given a state space $S = \{0, 1, 2, \ldots, N\}$ and a value $s \in \mathbb{Z}_+$ such that $s \neq a$, the Erlang CL model defined on this state space has stationary distribution π defined as follows:*

- *if $x < s$:*

$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1} \frac{a^j}{j!} + \frac{a^s}{s!} \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right]^{-1};$$

- *if $x \geq s$:*

$$\pi(x) = \left(\frac{a}{s}\right)^{x-s} \frac{a^s}{s!}\left[\sum_{j=0}^{s-1} \frac{a^j}{j!} + \frac{a^s}{s!} \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right]^{-1}.$$

for all $x \in S$.

2.2.2 Performance measures

As we did for Erlang C, we want to compute the performance measures for the Erlang CL model, i.e., the service level (SL), the average waiting time (ASA), and the occupancy (OCC). For this process, it is important to remember that the queue is finite, therefore the blocking probability plays an important role and the state space is $S = \{0, 1, \dots, N\}$.

Service Level

Consider first the service level of the model, which is again defined as in Definition 2.1.1. For Erlang CL systems, it is possible to compute the service level as stated in the following proposition.

Proposition 2.2.2. *Given the number of agents $s \in \mathbb{Z}_+$, the service level for non-blocked customers in Erlang CL systems can be written as*

$$\mathrm{SL} = \frac{1}{1-\pi(N)}\left[1 - \left(\sum_{i=0}^{N-s-1} \pi(s+i) \sum_{k=0}^{i} \frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} + \pi(N)\right)\right]$$

where $\pi(N)$ is the blocking probability.

Proof. Notice that when for instance $s+i$ customers are in the system, i of them are waiting in the queue. The rate by which from each state we can go down in the chain is $s\mu$ and we can leave each state after an exponential amount of time because the model is indeed a continuous-time Markov chain (CTMC). For this reason, when a new customer arrives in the system with i customers in the queue, in order to be served he has to wait for $i+1$ departures that occur each one with Exponential distribution with parameter $s\mu$. This means that the waiting time of customer i has a Gamma distribution with parameters $i+1$ and $s\mu$. Therefore we can compute the service level, i.e., the probability of waiting less than τ units of time by conditioning on the number of customers that are in the system.

Given X the random number of customers in the system, by definition we have that the service level for non-blocked customers is

$$\begin{aligned}
\mathbb{P}(W \le \tau) =& \frac{1}{1-\pi(N)}\left[1 - \left(\sum_{i=0}^{s-1} \mathbb{P}(W > \tau | X = i)\pi(i)\right.\right. \\
&\left.\left. + \sum_{i=0}^{N-s-1} \mathbb{P}(W > \tau | X = s+i)\pi(s+i) + \pi(N)\right)\right] \\
=& \frac{1}{1-\pi(N)}\left[1 - \left(\sum_{i=0}^{N-s-1} \mathbb{P}(\mathrm{Gamma}(i+1, s\mu) > \tau)\pi(s+i) + \pi(N)\right)\right]
\end{aligned}$$

where $\mathbb{P}(W > \tau | X = i) = 0$ for $i \in \{0, \dots, s-1\}$ because not all the agents are occupied and this means that a new customer does not have to wait before getting served. Notice also that the second summation is up to $N - s - 1$, because by the PASTA principle, the N-th customer does not have to wait, therefore in state N the service level is 1. Notice also that we want to divide by $1 - \pi(N)$ because we are interested in the customers that actually are accepted in the system and are not blocked.

In order to complete this computation we need to find both the probabilities of these Gamma random variables and the blocking probability. Notice that the probability $\pi(N)$ can be simply derived by the stationary distribution defined in the previous section. Regarding the Gamma random variables, we can explicitly write the cumulative function

$$\sum_{i=0}^{N-s-1} \mathbb{P}(\text{Gamma}(i+1, s\mu) > \tau)\pi(s+i) = \sum_{i=0}^{N-s-1} \pi(s+i) \int_\tau^\infty \frac{(\mu s)^{i+1}}{\Gamma(i+1)} x^i e^{-\mu s x} dx$$

and we can compute separately the integral by parts as follows:

$$\begin{aligned}
\int_\tau^\infty \frac{(\mu s)^{i+1}}{\Gamma(i+1)} x^i e^{-\mu s x} dx &= \int_\tau^\infty \frac{(\mu s x)^i}{i!} \cdot \mu s e^{-\mu s x} dx \\
&= -\int_\tau^\infty \frac{(\mu s x)^i}{i!} \cdot (-\mu s) e^{-\mu s x} dx \\
&= \frac{1}{i!} \left[(\mu s x)^i e^{-\mu s x} \Big|_\tau^\infty - \int_\tau^\infty i \cdot (\mu s x)^{i-1} \mu s e^{-\mu s x} dx \right] \\
&= \frac{(\mu s \tau)^i e^{-\mu s \tau}}{i!} + \frac{1}{(i-1)!} \left[(\mu s x)^{i-1} e^{-\mu s x} \Big|_\tau^\infty \right. \\
&\quad \left. - \int_\tau^\infty (i-1) \cdot (\mu s x)^{i-2} \mu s e^{-\mu s x} dx \right] \\
&= \frac{(\mu s \tau)^i e^{-\mu s \tau}}{i!} + \frac{(\mu s \tau)^{i-1} e^{-\mu s \tau}}{(i-1)!} + \frac{1}{(i-2)!} \left[(\mu s x)^{i-2} e^{-\mu s x} \Big|_\tau^\infty \right. \\
&\quad \left. - \int_\tau^\infty (i-2) \cdot (\mu s x)^{i-3} \mu s e^{-\mu s x} dx \right] \\
&= \dots \\
&= \frac{(\mu s \tau)^i e^{-\mu s \tau}}{i!} + \frac{(\mu s \tau)^{i-1} e^{-\mu s \tau}}{(i-1)!} + \dots + e^{-\mu s \tau} \\
&= \sum_{k=0}^{i} \frac{(\mu s \tau)^k e^{-\mu s \tau}}{k!}.
\end{aligned}$$

Then the service level is the following:

$$\begin{aligned}\mathbb{P}(W \leq \tau) &= \frac{1}{1-\pi(N)}\left[1 - \left(\sum_{i=0}^{N-s-1} \mathbb{P}(\text{Gamma}(i+1, s\mu) > \tau)\pi(s+i) + \pi(N)\right)\right] \\ &= \frac{1}{1-\pi(N)}\left[1 - \left(\sum_{i=0}^{N-s-1} \pi(s+i)\sum_{k=0}^{i}\frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} + \pi(N)\right)\right]\end{aligned}$$

which is what we wanted to prove. □

Average waiting time

The average waiting time is the second performance measure of our interest in the Erlang CL model.

Proposition 2.2.3. *Given the number of agents $s \in \mathbb{Z}_+$, the average speed of answer in Erlang CL systems can be written as*

$$\text{ASA} = \frac{1}{1-\pi(N)}\sum_{i=0}^{N-s-1}\frac{i+1}{s\mu}\pi(s+i)$$

where $\pi(N)$ is the blocking probability.

Proof. As already mentioned, since the distribution of $W|W>0$ is Gamma with parameters $i+1$ and $s\mu$, we have that $\mathbb{E}(W|W>0) = \frac{i+1}{s\mu}$. Therefore, it is possible to compute the performance measure as follows:

$$\begin{aligned}\mathbb{E}(W) &= \mathbb{E}(W|W>0)\mathbb{P}(W>0) + \mathbb{E}(W|W=0)\mathbb{P}(W=0) \\ &= \sum_{i=0}^{N-s-1}\frac{i+1}{s\mu}\pi(s+i)\end{aligned}$$

where indeed the waiting time is equal to zero whenever there are agents available to serve customers that enter the system. Notice also that again by the PASTA principle, the probability $\pi(N)$ represents the long-run stationary distribution of being in the state N, i.e., having N customers in the system. This leads to a zero value of the waiting time in this state because new arrivals are rejected. Moreover, we are interested in the waiting time for customers that have been accepted into the system, which means that the formula we want to compute is the following:

$$\mathbb{E}(W) = \frac{1}{1-\pi(N)}\sum_{i=0}^{N-s-1}\frac{i+1}{s\mu}\pi(s+i)$$

which is what we wanted to prove. □

Occupancy

The last performance measure that needs to be considered is the occupancy of a single server. Differently from the Erlang C model, in this case, the queue of the system has a finite capacity, therefore the shape of the occupancy has to change

$$\text{OCC} = \frac{\lambda}{s\mu}(1 - \pi(N))$$

where, by considering the term $1 - \pi(N)$, we are computing the occupancy of a single server when customers are accepted in the system, i.e., when they are not blocked.

2.3 The Erlang X model

The last Erlang model that we consider for this project is the Erlang X system. This model is a further extension of the systems presented so far because it has a finite capacity queue as the Erlang CL model and this means that also in Erlang X systems customers are rejected if there are already N persons in the system. Therefore, we have again a blocking probability that we need to take into account to study this model. Besides, the state space of the Markov chain that describes this model is again $S = \{0, 1, \ldots, N\}$.

However, it presents some new features: customers' patience and retrials. In a more realistic and precise description of these systems, abandonment plays an important role in the study of clients' behavior, since in real life when customers have to wait, they might want to leave the system. This means that the patience of customers needs to be analyzed and considered in the study of call center systems. In particular, for Erlang X models customers' patience is considered to follow an Exponential distribution with parameter γ. Moreover, in this situation, customers leave the system because either they have been served or because they do not want to wait in the queue anymore.

The second important new feature concerns retrials in the system. In particular, a fraction α of customers that abandon the system before they get served is willing to enter again the system and the remaining $1 - \alpha$ leaves forever. Besides, the fraction α is not included in the arrival rate λ, therefore the current implementation used by CCmath for managing this feature of the Erlang X model consists of a certain function that is used in the computation of every performance measure whenever there are retrials. Indeed, if $\alpha > 0$ the arrival rate λ is updated by this function creating the new arrival rate $\tilde{\lambda}$. In particular, given a small $\epsilon > 0$ the

algorithm is

$$\begin{cases} \lambda_1 = \lambda \\ \lambda_{k+1} = \lambda + \lambda_k \cdot \alpha \cdot \mathbb{P}_{ab}(\lambda_k) \\ \text{STOP if } |\frac{\lambda_{k+1}-\lambda_k}{\lambda_k}| \leq \epsilon \end{cases}$$

where we keep updating the abandonment probability with the current value of λ_k. $\mathbb{P}(\lambda_k)$ is the probability of abandonment that is updated at every step for the current value of λ_k. This probability is explained in more detail later in this chapter. At each step, the arrival rate augments a certain increment as long as the difference between two consecutive values is greater than ϵ.

This function is applied if and only if $\alpha > 0$, otherwise the arrival rate used for computing the performance measures remains λ. For this reason, in the following formulas for the Erlang X model, we will not consider redials, since they can be included in the system in a second moment outside of these computations.

2.3.1 Stationary distribution

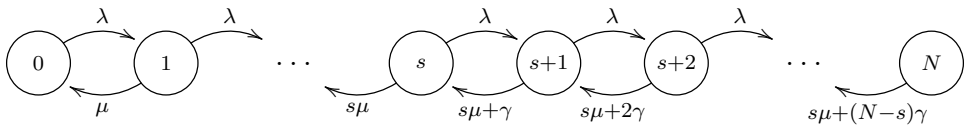

Figure 2.3.1: Markov chain of Erlang X model, where λ is the arrival rate, μ the service rate and γ is the abandonment rate. The maximum capacity of the system is N.

In Figure 2.3.1, the Markov chain that describes the Erlang X model is presented. We can see a similar behavior compared to the Erlang CL Markov chain with the extension of the parameter γ. Indeed, in this case, the rate of going down in the chain has changed because customers leave the system either when they get served or when they are not willing to stay in the queue anymore. Therefore, the CTMC stays in state $s + i$ for an Exponential amount of time with rate $\mu s + \gamma i$, when all the servers are full and there are i clients in the queue: there are s agents working at the same time with rate μ and there are i clients in the queue that have patience with rate γ. On the other hand, when the number of customers in the system is less than the number of servers, we have the same stationary distribution we found already for Erlang C and Erlang CL, since as long as there are servers available there is no abandonment because customers can be served immediately. Precisely, the stationary distribution computed by using the *Balance Equation* is the following:

- if $x < s$:

$$\begin{aligned}\lambda\pi(x) &= \mu(x+1)\pi(x+1)\\ \pi(x+1) &= \frac{\lambda}{\mu}\frac{1}{x+1}\pi(x)\\ \implies \pi(x) &= \frac{a^x}{x!}\pi(0);\end{aligned}$$

- if $s \le x \le N$:

$$\begin{aligned}\lambda\pi(x) &= (\mu s + (x+1-s)\gamma)\pi(x+1)\\ \pi(x+1) &= \frac{\lambda}{\mu s + (x+1-s)\gamma}\pi(x)\\ \implies \pi(x) &= \frac{\lambda^{x-s}}{[\mu s + (x-s)\gamma][\mu s + (x-1-s)\gamma]\cdot\cdots\cdot[\mu s+\gamma]}\pi(s)\\ &= \frac{\lambda^{x-s}}{\gamma^{x-s}[\frac{s}{b}+x-s][\frac{s}{b}+x-1-s]\cdot\cdots\cdot[\frac{s}{b}+1]}\pi(s)\\ &= \left(\frac{a}{b}\right)^{x-s}\frac{(\frac{s}{b})!}{(\frac{s}{b}+x-s)!}\pi(s)\\ &= \left(\frac{a}{b}\right)^{x-s}\frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+x-s)}\pi(s)\\ &= \left(\frac{a}{b}\right)^{x-s}\frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+x-s)}\frac{a^s}{s!}\pi(0)\end{aligned}$$

where we define $b = \frac{\gamma}{\mu}$. We know that $\sum_{x=0}^{N}\pi(x) = 1$, then by normalization we obtain the following:

$$\begin{aligned}\sum_{x=0}^{N}\pi(x) &= \sum_{x=0}^{s-1}\frac{a^x}{x!}\pi(0) + \frac{a^s}{s!}\pi(0) + \sum_{j=s+1}^{N}\left(\frac{a}{b}\right)^{j-s}\frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+j-s)}\frac{a^s}{s!}\pi(0)\\ &= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\sum_{x=0}^{N-s}\left(\frac{a}{b}\right)^{x}\frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+x)}\right]\\ \implies \pi(0) &= \left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\sum_{x=0}^{N-s}\left(\frac{a}{b}\right)^{x}\frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+x)}\right]^{-1}.\end{aligned}$$

Therefore, the stationary distribution of the Erlang X model is presented in the following proposition.

Proposition 2.3.1. *Given a state space $S = \{0, 1, 2, \ldots, N\}$ and a value $s \in \mathbb{Z}_+$, the Erlang X model defined on this state space has stationary distribution π defined as follows:*

- *if* $x < s$*:*

$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\sum_{j=0}^{N-s}\left(\frac{a}{b}\right)^j \frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+j)}\right]^{-1};$$

- *if* $s \leq x \leq N$*:*

$$\pi(x) = \left(\frac{a}{b}\right)^{x-s}\frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+x-s)}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\sum_{j=0}^{N-s}\left(\frac{a}{b}\right)^j \frac{\Gamma(1+\frac{s}{b})}{\Gamma(\frac{s}{b}+1+j)}\right]^{-1};$$

for all $x \in S$*.*

2.3.2 Performance measures

In addition to the performance measures we already discussed for Erlang C and Erlang CL, in this case, there is a new measure that can give great insights and information regarding the behavior of this model: the *probability of abandonment*.

Service Level

Starting again with the service level, observe that for this model we will discuss this performance measure from different perspectives. There will be presented three different types of service levels in this section: *virtual*, *answered*, and *offered*.

What distinguishes the three types of SL is the abandonment behavior, i.e., the random variable $G \sim Exp(\gamma)$ that represents the patience of a single client. Notice first that the service level is a quotient and therefore we have to consider as a numerator the number of served calls before the acceptable waiting time τ because in all cases we are interested in computing the percentage of served calls among different categories of calls. In particular, the *virtual* service level can be defined as

$$SL_v = \mathbb{P}(W \leq \tau)$$

where the quantity of interest is the probability that the waiting time of a single customer is less than τ assuming that he does not leave the system and stays until he gets served. On the other hand, the *answered* service level is

$$SL_a = \frac{\#(\text{served} \leq \tau)}{\#(\text{served})} = \frac{\mathbb{P}(W \leq \tau, W < G)}{\mathbb{P}(W < G)}$$

where the percentage of answered calls is given by the probability that the waiting time W of a single customer is less than his patience, i.e., there is no abandonment.

Finally, the *offered* service level is

$$SL_o = \frac{\#(\text{served} \leq \tau)}{\#(\text{offered})} = \mathbb{P}(W \leq \tau, W < G)$$

where indeed we want to compute the percentage of answered calls among all the calls received in the system.

Given these definitions, we can observe that the virtual service time does not consider the abandonment behavior of the call we are considering. To be more precise, this customer is considered to have infinite patience. The aim of the first definition is to compute the service level in case this customer is willing to wait in the queue until all other clients in front of him get served or leave the system.

This performance measure can be computed in all three cases, however, it will give different insights according to the perspective of the considered type of service.

Virtual Service Level

Consider first the definition of virtual service level. As already mentioned, when all servers are busy, the CTMC goes from state $s + i$ to state $s + i - 1$ after an exponential amount of time with rate $s\mu + \gamma i$. Notice that the rate changes depending on the state of the chain and for this reason, it is not possible to consider a Gamma distribution anymore for the waiting time. Therefore, we have to introduce another distribution that perfectly describes this stochastic process: the Hypoexponential distribution.

A Hypoexponential random variable X is defined as the sum of $X_1, \ldots, X_n$ independent random variables, where each X_i has Exponential distribution with parameter λ_i. By definition, the cumulative distribution function is the following:

$$\begin{aligned} F_X(\tau) &= 1 - S_X(\tau) \\ S_X(\tau) &= \sum_{i=1}^{n} W_{n,i} e^{-\lambda_i \tau} \\ W_{n,i} &= \prod_{j \neq i} \frac{\lambda_j}{\lambda_j - \lambda_i}. \end{aligned}$$

Therefore we can compute the following quantity:

$$\begin{aligned}\mathbb{P}(W > \tau) &= \sum_{i=0}^{N-s-1} \mathbb{P}(W > \tau | X = s+i)\pi(s+i) \\ &= \sum_{i=0}^{N-s-1} \mathbb{P}(\text{Hypoexponential}(s\mu + \gamma i) > \tau)\pi(s+i) \\ &= \sum_{i=0}^{N-s-1} \pi(s+i) \sum_{j=0}^{i} W_{i,j} e^{-\lambda_j \tau}\end{aligned}$$

where we sum to $N-s-1$ customers in the queue. For the PASTA principle, we have that the N-th arrival represents also the long-run probability of being in the state N, where we do not see a positive service level, because at that state new customers are rejected. This means that for the N-th term we consider the blocking probability $\pi(N)$. Therefore the probability that the waiting time of a customer that has been accepted in the queue is less than τ is the following:

$$\text{SL}_v = \mathbb{P}(W \leq \tau) = \frac{1 - \sum_{i=0}^{N-s-1} \mathbb{P}(\text{Hypoexponential}(s\mu + \gamma i) > \tau)\pi(s+i) - \pi(N)}{1 - \pi(N)}$$

where indeed we divide by $1-\pi(N)$ because we want to compute the service level of those customers that have been accepted in the system.

Furthermore, for the Erlang X model, we want to compute the terms $W_{i,j}$, where i is the number of customers in the queue when the chain is in state $s+i$. Notice that the rates are $\lambda_j = \mu s + \gamma j$ for $j = 0, \ldots, i$. Then we can compute the following:

$$\begin{aligned}W_{i,j} &= \prod_{k \neq j} \frac{\mu s + \gamma k}{\mu s + \gamma k - (\mu s + \gamma j)} \\ &= \prod_{k \neq j} \frac{\mu s + \gamma k}{\gamma k - \gamma j} \\ &= \prod_{k \neq j} \frac{\frac{s}{b} + k}{k - j} \\ &= \left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right) \cdot \ldots \cdot \left(\frac{s}{b}+i\right) \cdot \frac{1}{\left(\frac{s}{b}+j\right)} \cdot \frac{1}{\prod_{k \neq j} k - j}\end{aligned}$$

where we want to rewrite $\frac{s}{b} = \frac{\mu s}{\gamma}$, because we defined $b = \frac{\gamma}{\mu}$. Furthermore, consider

also the following computation:

$$\begin{aligned}\prod_{k\neq j}(k-j) &= \prod_{k=0}^{j-1}(k-j)\cdot\prod_{k=j+1}^{i}(k-j)\\ &= \prod_{k=0}^{j-1}(-1)^j|k-j|\cdot\prod_{l=1}^{i-j}l\\ &= (-1)^j\prod_{k=0}^{j-1}|j-k|\cdot(i-j)!\\ &= (-1)^j\cdot j!\cdot(i-j)!\end{aligned}$$

which leads to the following expression for the terms $W_{i,j}$:

$$W_{i,j} = \left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+i\right)\cdot\frac{1}{\left(\frac{s}{b}+j\right)}\cdot\frac{(-1)^j}{j!\cdot(i-j)!}.$$

Finally, we are able to compute the cumulative distribution function for the Erlang X model as follows:

$$\begin{aligned}S_X(\tau) &= \sum_{j=1}^{i}W_{i,j}e^{-\gamma\tau(\frac{s}{b}+j)}\\ &= \sum_{j=1}^{i}\left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+i\right)\cdot\frac{1}{\left(\frac{s}{b}+j\right)}\cdot\frac{(-1)^j}{j!\cdot(i-j)!}e^{-\gamma\tau(\frac{s}{b}+j)}\\ &= \left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+i\right)\sum_{j=1}^{i}\frac{1}{\left(\frac{s}{b}+j\right)}\cdot\frac{(-1)^j}{j!\cdot(i-j)!}e^{-\gamma\tau(\frac{s}{b}+j)}.\end{aligned}$$

Therefore the term for the service level with the Hypoexponential distribution can

be computed as follows:

$$\begin{aligned}
&\sum_{i=0}^{N-s-1} \mathbb{P}(W > \tau | X = s+i)\pi(s+i) \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) \cdot \left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right) \cdot \cdots \cdot \left(\frac{s}{b}+i\right) \sum_{j=1}^{i} \frac{1}{\frac{s}{b}+j} \cdot \frac{(-1)^j}{j! \cdot (i-j)!} e^{-\gamma\tau(\frac{s}{b}+j)} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) \cdot \left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right) \cdot \cdots \cdot \left(\frac{s}{b}+i\right) \frac{1}{i!} \sum_{j=1}^{i} (-1)^j \frac{i!}{j! \cdot (i-j)!} \frac{e^{-\gamma\tau(\frac{s}{b}+j)}}{\frac{s}{b}+j} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) \cdot \left(\frac{s}{b}\right)\left(\frac{s}{b}+1\right) \cdot \cdots \cdot \left(\frac{s}{b}+i\right) \frac{1}{i!} \sum_{j=1}^{i} (-1)^j \binom{i}{j} \frac{e^{-\gamma\tau(\frac{s}{b}+j)}}{\frac{s}{b}+j}
\end{aligned}$$

which is the definition of the virtual service level suggested in Lemma 1 of article [4]. This Lemma provides also a second definition for the virtual service level:

$$\begin{aligned}
&\sum_{i=0}^{N-s-1} \mathbb{P}(W > \tau | X = s+i)\pi(s+i) \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\gamma\tau} \frac{\frac{s}{b}(\frac{s}{b}+1) \cdot \cdots \cdot (\frac{s}{b}+i)}{i!} \sum_{j=0}^{i} (-1)^j \binom{i}{j} \frac{e^{-\gamma\tau(\frac{s}{b}+j)}}{\frac{s}{b}+j} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-s\mu\tau} \sum_{j=0}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1) \cdot \cdots \cdot (\frac{s}{b}+j-1)(1-e^{-\gamma\tau})^j}{j!}
\end{aligned}$$

where the term $\frac{s}{b}(\frac{s}{b}+1) \cdot \cdots \cdot (\frac{s}{b}+j-1)$ for $j = 0$ is assumed to be equal to 1. The second expression of this formula is preferable for computations since the term $(-1)^j$ in the sum does not need to be calculated and therefore all terms have the same sign. In order to utilize the second formula for the computation, it is necessary to prove the equality between the two expressions A and B:

$$\begin{aligned}
A &= \frac{\frac{s}{b}(\frac{s}{b}+1) \cdot \cdots \cdot (\frac{s}{b}+i)}{i!} \sum_{j=0}^{i} (-1)^j \binom{i}{j} \frac{e^{-\gamma\tau(\frac{s}{b}+j)}}{\frac{s}{b}+j} \\
B &= e^{-s\mu\tau} \sum_{j=0}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1) \cdot \cdots \cdot (\frac{s}{b}+j-1)(1-e^{-\gamma\tau})^j}{j!}.
\end{aligned}$$

Fact.

$$A = B$$

where A and B are defined as before.

Proof. Consider first the expression B and apply the Binomial theorem on $(1 - e^{-\gamma\tau})^j$ as follows:

$$\begin{aligned} B &= e^{-s\mu\tau} \sum_{j=0}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+j-1)(1-e^{-\gamma\tau})^j}{j!} \\ &= e^{-s\mu\tau} \sum_{j=0}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+j-1)}{j!} \sum_{k=0}^{j} \binom{j}{k} (-1)^k e^{-\gamma\tau k}. \end{aligned}$$

Notice that the indices of the summations are in this order: $0 < k < j < i$. Therefore, it is possible to rewrite this expression as follows:

$$\begin{aligned} B &= e^{-s\mu\tau} \sum_{k=0}^{i} \sum_{j=k}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+j-1)}{j!} \cdot \frac{j!}{k!(j-k)!} (-1)^k e^{-\gamma\tau k} \\ &= e^{-s\mu\tau} \sum_{k=0}^{i} \frac{(-1)^k e^{-\gamma\tau k}}{k!} \sum_{j=k}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+j-1)}{(j-k)!}. \end{aligned}$$

By renaming the sum over j as C, we can compute it as follows:

$$\begin{aligned} C &= \sum_{j=k}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+j-1)}{(j-k)!} \\ &= \sum_{l=0}^{i-k} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+l+k-1)}{l!} \\ &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+k-1\right) \sum_{l=0}^{i-k} \frac{(\frac{s}{b}+k)\cdot\dots\cdot(\frac{s}{b}+k-1+l)}{l!} \end{aligned}$$

where we made the substitution $j = l + k$. Later, we can consider the following factorial values

$$\begin{aligned} C &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+k-1\right) \sum_{l=0}^{i-k} \frac{(\frac{s}{b}+k-1)!(\frac{s}{b}+k)\cdot\dots\cdot(\frac{s}{b}+k-1+l)}{(\frac{s}{b}+k-1)!\cdot l!} \\ &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+k-1\right) \sum_{l=0}^{i-k} \frac{(\frac{s}{b}+k-1+l)!}{(\frac{s}{b}+k-1)!\cdot l!} \\ &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\dots\cdot\left(\frac{s}{b}+k-1\right) \sum_{l=0}^{i-k} \binom{\frac{s}{b}+k-1+l}{l}. \end{aligned}$$

Recall now the following property of the binomial coefficient:

$$\sum_{i=0}^{n} \binom{k+i}{i} = \binom{k+n+1}{n}$$

that we can apply as follows

$$\begin{aligned} C &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\cdots\cdot\left(\frac{s}{b}+k-1\right)\sum_{l=0}^{i-k}\binom{\frac{s}{b}+k-1+l}{l} \\ &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\cdots\cdot\left(\frac{s}{b}+k-1\right)\binom{\frac{s}{b}+k-1+i-k+1}{i-k} \\ &= \frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\cdots\cdot\left(\frac{s}{b}+k-1\right)\binom{\frac{s}{b}+i}{i-k}. \end{aligned}$$

Now, by substituting the last expression of C in B, we obtain:

$$B = e^{-s\mu\tau}\sum_{k=0}^{i}\frac{(-1)^k e^{-\gamma\tau k}}{k!}\frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\cdots\cdot\left(\frac{s}{b}+k-1\right)\binom{\frac{s}{b}+i}{i-k}.$$

If we explicit the binomial coefficient, we obtain

$$\begin{aligned} B &= e^{-s\mu\tau}\sum_{k=0}^{i}\frac{(-1)^k e^{-\gamma\tau k}}{k!}\frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\cdots\cdot\left(\frac{s}{b}+k-1\right)\frac{(\frac{s}{b}+i)!}{(i-k)!(\frac{s}{b}+k)!} \\ &= e^{-s\mu\tau}\sum_{k=0}^{i}\frac{(-1)^k e^{-\gamma\tau k}}{k!}\frac{s}{b}\left(\frac{s}{b}+1\right)\cdot\cdots\cdot\left(\frac{s}{b}+k-1\right)\cdot\frac{(\frac{s}{b}-1)!\frac{s}{b}(\frac{s}{b}+1)\cdot\cdots\cdot(\frac{s}{b}+i)}{(i-k)!(\frac{s}{b}+k-1)!(\frac{s}{b}+k)} \\ &= \sum_{k=0}^{i}\frac{(-1)^k e^{-\gamma\tau(\frac{s}{b}+k)}}{i!}\cdot\frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\cdots\cdot(\frac{s}{b}+i)}{(\frac{s}{b}+k)}\cdot\frac{i!}{(i-k)!\cdot k!} \end{aligned}$$

where we used the fact that $\frac{s}{b}(\frac{s}{b}+1)\cdot\cdots\cdot(\frac{s}{b}+k-1)\cdot(\frac{s}{b}-1)! = (\frac{s}{b}+k-1)!$, therefore it simplifies. Finally, we can rewrite this last expression as follows:

$$\begin{aligned} B &= \sum_{k=0}^{i}\frac{(-1)^k e^{-\gamma\tau(\frac{s}{b}+k)}}{i!}\cdot\frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\cdots\cdot(\frac{s}{b}+i)}{(\frac{s}{b}+k)}\cdot\binom{i}{k} \\ &= \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\cdots\cdot(\frac{s}{b}+i)}{i!}\sum_{k=0}^{i}(-1)^k\cdot\binom{i}{k}\frac{e^{-\gamma\tau(\frac{s}{b}+k)}}{\frac{s}{b}+k} \end{aligned}$$

which is exactly the expression A that we wanted to find.
This proves that $A = B$. □

We proved indeed the existence of the second expression for the virtual service level, which turns out to be preferable in terms of computation as already mentioned. Therefore, the formula we are going to use is the following:

$$\mathrm{SL}_v = \mathbb{P}(W \leq \tau) = \frac{1 - \sum_{i=0}^{N-s-1} \mathbb{P}(\text{Hypoexponential}(s\mu + \gamma i) > \tau)\pi(s+i) - \pi(N)}{1 - \pi(N)}$$

where

$$\begin{aligned} &\sum_{i=0}^{N-s-1} \mathbb{P}(\text{Hypoexponential}(s\mu + \gamma i) > \tau)\pi(s+i) \\ &= \sum_{i=0}^{N-s-1} \pi(s+i)e^{-s\mu\tau} \sum_{j=0}^{i} \frac{\frac{s}{b}(\frac{s}{b}+1)\cdot\dots\cdot(\frac{s}{b}+j-1)(1-e^{-\gamma\tau})^j}{j!} \end{aligned}$$

Answered Service Level

The answered service level represents the number of calls that have been answered before the acceptable waiting time τ among all the connected calls. As mentioned before, we want to compute the following formula:

$$\mathrm{SL}_a = \frac{\mathbb{P}(W \leq \tau, W < G)}{\mathbb{P}(W < G)}$$

where the event $\{W < G\}$ is exactly the complementary of the abandonment event $\{G \leq W\}$, where indeed the patience of a single customer is less than the waiting time and therefore he leaves the system. This means that we can compute the following:

$$\mathbb{P}(W < G) = 1 - \mathbb{P}_{ab}$$

where $\mathbb{P}_{ab}$ is the probability of abandonment, another interesting performance measure that is described later in this chapter. Therefore, what we still need to compute is the numerator of the definition of SL_a and we can do it as follows:

$$\begin{aligned} \mathbb{P}(W \leq \tau, W < G) &= 1 - \mathbb{P}(W > \tau, W < G) \\ &= 1 - \sum_{k=0}^{N-s-1} \mathbb{P}(W > \tau, W < G | X = s+k)\pi(s+k) \\ &= 1 - \sum_{k=0}^{N-s-1} (1 - \mathbb{P}(W \leq \tau, W < G | X = s+k))\pi(s+k) \end{aligned}$$

where the random variable X represents the number of customers in the system and we are summing over the customers that are in the queue. In order to compute each

of these terms, we can define the function $G_k(\tau) = \mathbb{P}(W \leq \tau, W < G | X = s + k)$, therefore

$$\mathbb{P}(W \leq \tau, W < G) = 1 - \sum_{k=0}^{N-s-1} (1 - G_k(\tau))\pi(s + k).$$

The expression of the function G_k comes from the computations used at CCmath for the implementation of both the answered and offered service levels. The used expression is the following:

$$G_k(\tau) = d_k - \sum_{i=0}^{k} c_{i,k} e^{-(\mu s + i\gamma + \tilde{\gamma})\tau}$$

where $\tilde{\gamma}$ is the abandonment rate of the customer arriving when k customers are already in the queue and they have instead abandonment rate γ. The reason behind this choice is that when $\tilde{\gamma} = 0$, we can actually compute the virtual waiting time with this formula, whereas if $\tilde{\gamma}$ is a non-zero value, then this function is helpful to compute other types of service level.

Also, the terms d_k and $c_{i,k}$ are defined as follows:

$$\begin{aligned} d_k &= \frac{s\mu + k\gamma}{s\mu + k\gamma + \tilde{\gamma}} d_{k-1} \\ c_{i,k} &= \frac{s\mu + k\gamma}{(k - i)\gamma} c_{i,k-1} \\ c_{k,k} &= d_k - \sum_{i=0}^{k-1} c_{i,k} \end{aligned}$$

with first terms

$$d_0 = c_{0,0} = \frac{s\mu}{s\mu + \tilde{\gamma}}.$$

In conclusion, by using all these formulas it is possible to compute the answered service level as follows:

$$\mathrm{SL}_a = \frac{1 - \sum_{k=0}^{N-s-1} (1 - G_k(\tau))\pi(s + k)}{(1 - \mathbb{P}_{ab})(1 - \pi(N))}$$

where we divide by the term $1 - \pi(N)$ because we are interested in computing this service level only for those customers that have been accepted in the system.

Offered Service Level

The last service level we want to compute is the offered service level. In this case, we have the following formula:

$$\mathrm{SL}_o = \mathbb{P}(W \leq \tau, W < G)$$

where this probability can be computed as for the answered service level by the function G_k as follows:

$$\begin{aligned}
\mathrm{SL}_o &= \mathbb{P}(W \leq \tau, W < G) \\
&= 1 - \mathbb{P}(W > \tau, W < G) \\
&= 1 - \sum_{k=0}^{N-s} \mathbb{P}(W > \tau, W < G | X = s+k)\pi(s+k) \\
&= 1 - \sum_{k=0}^{N-s} (1 - \mathbb{P}(W \leq \tau, W < G | X = s+k))\pi(s+k) \\
&= 1 - \sum_{k=0}^{N-s} (1 - G_k(\tau))\pi(s+k)
\end{aligned}$$

with G_k defined as before. Notice that for the offered service level we are summing until $N - s$ customers in the queue because in this case, we are interested in the service level offered in general and not only for the service level obtained by customers that have been accepted to the queue.

Average waiting Time

After analyzing the service level, the second performance measure of interest is the waiting time, also called the average speed of answer (ASA).

Proposition 2.3.2. *Given the number of agents $s \in \mathbb{Z}_+$, the average speed of answer in Erlang X systems can be written as*

$$\mathrm{ASA} = \frac{1}{1 - \pi(N)} \sum_{i=0}^{N-s-1} \frac{i+1}{s\mu + i\gamma} \pi(s+i)$$

where $\pi(N)$ is the blocking probability.

Proof. We know that for a Hypoexponential random variable $X = X_1 + \cdots + X_n$ where $X_i \sim Exp(\lambda_i)$ we have that

$$\mathbb{E}(X) = \frac{1}{\lambda_1} + \cdots + \frac{1}{\lambda_n}$$

In this case, we have that the waiting time $W|W > 0$ is a Hypoexponential random variable such that for each i customer in the queue, we have that the exponential

rate of leaving the system is $\lambda_i = s\mu + i\gamma$ for $i = 0, \ldots, N-s$. Therefore, we have the following formula:

$$\begin{aligned}\mathbb{E}(W) &= \mathbb{E}(W|W>0)\mathbb{P}(W>0) + \mathbb{E}(W|W=0)\mathbb{P}(W=0) \\ &= \sum_{i=0}^{N-s-1} \frac{i+1}{s\mu + i\gamma}\pi(s+i)\end{aligned}$$

where, by PASTA, the N-th customer arrival represents also the long-run probability of having N customers in the system, which means that the waiting time is zero because customers are not even allowed in the queue anymore. Therefore there is a blocking probability that has to play a role in this computation. In particular, if we want to compute the ASA for all customers that have not been rejected we have to divide the last expression by the probability $1 - \pi(N)$, which is exactly the probability of not being blocked. Therefore:

$$\mathbb{E}(W) = \frac{1}{1-\pi(N)} \sum_{i=0}^{N-s-1} \frac{i+1}{s\mu + i\gamma}\pi(s+i)$$

which is what we wanted to prove. □

Probability of Abandonment

As mentioned before, another interesting performance measure to analyze in the Erlang X model is certainly the probability of abandonment P_{ab} of customers that have been accepted in the system. By definition we have

$$\mathbb{P}_{ab} = \sum_{i=s+1}^{N} \pi(i)(i-s)\frac{\gamma}{\lambda}$$

which represents the fraction between the total number of abandonment transitions per time unit and the number of arrivals per time unit.

Occupancy

Also for the Erlang X model, it is interesting to analyze the behavior of the occupancy of the system. Recall that for this model we have to take into account blocking probability and abandonment.

The other important features of the Erlang X model that affect the occupancy of a single server are the blocking probability and the abandonment probability. When a customer is blocked, he cannot be served. Then, the probability $\pi(N)$ plays an important role because we are interested in considering all customers

that actually get served. For the same reason, it is important to consider also the probability of abandonment $\mathbb{P}_{ab}$ because in this case the customer leaves the system before he gets served, therefore we should consider all customers that do not abandon.

In conclusion, the formula for the occupancy of a single server in the Erlang X model is the following:

$$\mathrm{OCC} = \frac{\lambda}{s\mu}(1 - \pi(N) - \mathbb{P}_{ab}).$$

Chapter 3

Continuous models

The goal of this research consists of finding the optimal way to compute a fractional number of agents s when the value of a certain performance measure is given and vice-versa. The main problem that we have to face if we want to work with real values of s is that the Erlang systems defined in the previous chapter are described as Markov chains. As shown before, these queuing models use discrete formulas for the computation of the stationary distribution or the performance measures, in terms of the number of agents s. Therefore, the first goal consists of finding continuous functions of s that equally describe the Erlang systems.

In particular, during this research, we were able to find the equivalent continuous model for Erlang C and Erlang CL systems.

3.1 Continuous formulas for the Erlang C model

The simple structure of the Erlang C model, compared to Erlang CL and Erlang X, allows working easily on the formulas that describe its performance measures. Recall that in the previous chapter, we defined the stationary distribution of the Markov chain and the service level, the average waiting time, and the occupancy as performance measures. The goal is to rewrite them as continuous functions with respect to the number of agents s, in order to compute them even if s is a real value and not only an integer.

The following theorem defines a function $G(a, s)$ that has both continuous and discrete expressions. This function will play an essential role in the continuous computation of the stationary distribution and the performance measures in the Erlang C model.

Theorem 3.1.1. *Given $a = \frac{\lambda}{\mu}$, we can define the function $G(a, s)$ as follows*

$$G(a,s) = \sum_{x=0}^{s-1} \frac{s! a^{x-s}}{x!} = \int_0^\infty s e^{-at}(1+t)^{s-1} dt.$$

Proof. We want to prove the equality between the discrete and the continuous expressions of the function $G(a, s)$. First, recall the following equality of the Binomial theorem:

$$(1+t)^{s-1} = \sum_{x=0}^{s-1} \binom{s-1}{x} t^{s-1-x}.$$

Consider now the integral expression and by the Binomial theorem we obtain

$$\begin{aligned}
\int_0^\infty s e^{-at}(1+t)^{s-1} dt &= \int_0^\infty s e^{-at} \sum_{x=0}^{s-1} \binom{s-1}{x} t^{s-1-x} dt \\
&= \int_0^\infty s e^{-at} \sum_{x=0}^{s-1} \frac{(s-1)!}{x!(s-1-x)!} t^{s-1-x} dt \\
&= \sum_{x=0}^{s-1} \frac{s!}{x!(s-1-x)!} \int_0^\infty e^{-at} t^{s-1-x} dt \\
&= \sum_{x=0}^{s-1} \frac{s!}{x!(s-1-x)!} \int_0^\infty \frac{1}{a} e^{-u} \left(\frac{u}{a}\right)^{s-1-x} du \\
&= \sum_{x=0}^{s-1} \frac{s! a^{s-x}}{x!(s-1-x)!} \int_0^\infty e^{-u} u^{s-1-x} du \\
&= \sum_{x=0}^{s-1} \frac{s! a^{s-x}}{x!(s-1-x)!} \cdot \Gamma(s-x) \\
&= \sum_{x=0}^{s-1} \frac{s! a^{s-x}}{x!(s-1-x)!} \cdot (s-1-x)! \\
&= \sum_{x=0}^{s-1} \frac{s! a^{s-x}}{x!}
\end{aligned}$$

where we can exchange the integral and the summation because the latter is finite. Moreover, we change the variable in the integral as $t = \frac{u}{a}$ and we use the definition of Gamma function:

$$\Gamma(z) = \int_0^{+\infty} t^{z-1} e^{-t} dt$$

where in this case $z = s - x$. Finally, we use the fact that a Gamma function can be rewritten as a factorial number: $\Gamma(z) = (z-1)!$. By using simple properties of analysis, we proved the equality between the continuous and the discrete expression for the function $G(a, s)$. □

After proving the expressions of the function $G(a, s)$, we can find how to rewrite the stationary distribution and the performance measures as functions of $G(a, s)$.

3.1.1 Continuous stationary distribution

Recall from the previous chapter that the stationary distribution for the Erlang C model is the following:

- if $x < s$:

$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1};$$

- if $x \geq s$:

$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1}.$$

The goal is to rewrite these formulas as functions of $G(a, s)$. Notice first that the discrete sum appears in the term in brackets for both cases. Consider only that term: by multiplying and dividing by $\frac{a^s}{s!}$ we obtain the following:

$$\begin{aligned}\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)} &= \frac{a^s}{s!}\left(\sum_{j=0}^{s-1}\frac{a^{x-s}s!}{j!} + \frac{a^s s!}{(s-1)!(s-a)}\right)\\ &= \frac{a^s}{s!}\left(G(a,s) + \frac{s!}{(s-1)!(s-a)}\right)\\ &= \frac{a^s}{s!}\left(G(a,s) + \frac{s}{s-a}\right).\end{aligned}$$

Recall that in Erlang C systems we have the assumption $a < s$, therefore we can rewrite the stationary distribution as follows:

Proposition 3.1.2. *Given a state space $S = \{0, 1, 2, \dots\}$ and a value $s \in \mathbb{Z}_+$ such that $s > a = \frac{\lambda}{\mu}$, the Erlang C model defined on this state space has stationary distribution π defined as follows:*

- *if* $x < s$*:*

$$\pi(x) = \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{s}{s-a}\right)\right]^{-1};$$

- *if* $x \geq s$*:*

$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\left[G(a,s) + \frac{s}{s-a}\right]^{-1};$$

where $G(a,s)$ *is defined as in Theorem 3.1.1.*

In both cases, we have continuous functions with respect to s, because of the composition of continuous functions.

3.1.2 Continuous performance measures

After the stationary distribution, we are also interested in finding new formulations for the performance measures of Erlang C as functions of $G(a,s)$.

Proposition 3.1.3. *In Erlang C systems, the probability of delay can be written as*

$$C(s,a) = \frac{1}{1 + G(a,s)\cdot(1-\frac{a}{s})}$$

where $G(a,s)$ *is defined as in Theorem 3.1.1.*

Proof. Recall the formula for the probability of delay of Erlang C models given in Proposition 2.1.3. Then, we can write with simple computations that

$$\begin{aligned}
C(a,s) &= \frac{a^s}{(s-1)!(s-a)}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}\right]^{-1} \\
&= \frac{a^s}{(s-1)!(s-a)} \cdot \frac{1}{\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)}} \\
&= \frac{1}{1 + \sum_{j=0}^{s-1}\frac{a^{j-s}}{j!}\cdot(s-1)!(s-a)} \\
&= \frac{1}{1 + \sum_{j=0}^{s-1}\frac{a^{j-s}}{j!}\cdot\frac{s!(s-a)}{s}} \\
&= \frac{1}{1 + \sum_{j=0}^{s-1}\frac{a^{j-s}s!}{j!}\cdot(1-\frac{a}{s})} \\
&= \frac{1}{1 + G(a,s)\cdot(1-\frac{a}{s})}
\end{aligned}$$

which is what we wanted to prove. □

Proposition 3.1.4. *In Erlang C systems, the service level can be written as*

$$\mathrm{SL} = 1 - \frac{e^{-(s\mu-\lambda)\tau}}{1 + G(a,s)\cdot(1-\frac{a}{s})}$$

where $G(a,s)$ is defined as in Theorem 3.1.1.

Proof. Recall Proposition 2.1.2 which states that the service level in Erlang C systems can be computed as

$$\mathrm{SL} = 1 - C(s,a)e^{-(s\mu-\lambda)\tau}.$$

By using the expression of $C(s,a)$ given in Proposition 3.1.3, we find the following:

$$\begin{aligned} P(W \leq \tau) &= 1 - C(a,s)e^{-(s\mu-\lambda)\tau} \\ &= 1 - \frac{e^{-(s\mu-\lambda)\tau}}{1 + G(a,s)\cdot(1-\frac{a}{s})} \end{aligned}$$

which is what we wanted to prove. □

Proposition 3.1.5. *In Erlang C systems, the average speed of answer can be written as*

$$\mathrm{ASA} = \frac{1}{s\mu(1-\frac{a}{s})(1 + G(a,s)\cdot(1-\frac{a}{s}))}$$

where $G(a,s)$ is defined as in Theorem 3.1.1.

Proof. Recall the formulation of ASA given in Proposition 2.1.4

$$\mathrm{ASA} = \frac{C(a,s)}{s\mu-\lambda} = \frac{C(a,s)\frac{1}{\mu}}{s-a}.$$

By using the expression of $C(s,a)$ given in Proposition 3.1.3, we can compute

$$\begin{aligned} E(W) &= \frac{C(a,s)\frac{1}{\mu}}{s-a} \\ &= \frac{1}{1 + G(a,s)\cdot(1-\frac{a}{s})} \cdot \frac{\frac{1}{\mu}}{s(1-\frac{a}{s})} \\ &= \frac{1}{s\mu(1-\frac{a}{s})(1 + G(a,s)\cdot(1-\frac{a}{s}))} \end{aligned}$$

which is what we wanted to prove. □

Regarding occupancy, since the expression of this performance measure for the Erlang C model is already a continuous function with respect to the number of agents s, we do not need to make any modifications.

3.2 Continuous formulas for the Erlang CL model

After providing continuous formulas for both the stationary distribution and the performance measures of the Erlang C model, we want to achieve similar results for Erlang CL systems. The idea is to follow the same arguments and therefore we are still considering the important result stated in Theorem 3.1.1.

3.2.1 Continuous stationary distribution

Recall from the previous chapter that a Markov chain in the Erlang CL model follows this stationary distribution:

- if $x < s$:
$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right]^{-1};$$
- if $x \geq s$:
$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right]^{-1}.$$

for all $x \in S = \{0, 1, \ldots, N\}$.

Notice that by using the result of Theorem 3.1.1, we can rewrite the term between brackets as follows:

$$\begin{aligned}\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}} &= \frac{a^s}{s!}\left(\sum_{j=0}^{s-1}\frac{a^{j-s}s!}{j!} + \frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right)\\ &= \frac{a^s}{s!}\left(G(a,s) + \frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right)\end{aligned}$$

which is a composition of continuous functions with respect to the number of agents s. Notice that also for the continuous formulas of Erlang CL, we analyze the particular case when $a = s$ in Appendix A.

Therefore, we can rewrite the continuous formulation of the stationary distribution as follows:

Proposition 3.2.1. *Given a state space $S = \{0, 1, 2, \ldots, N\}$ and a value $s \in \mathbb{Z}_+$ such that $s \neq a$, the Erlang CL model defined on this state space has stationary distribution π defined as follows:*

- *if $x < s$:*
$$\pi(x) = \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right)\right]^{-1};$$

- *if* $x \geq s$*:*

$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\left[G(a,s) + \frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right]^{-1};$$

for all $x \in S = \{0, 1, \ldots, N\}$*, where* $G(a,s)$ *is defined as in Theorem 3.1.1.*

3.2.2 Continuous performance measures

We are also interested in finding continuous formulas for the performance measures in Erlang CL systems in order to compute these features of the model by using a fractional number of agents s.

Notice that in this section we discuss only the general case when $s \neq a$, because for $s = a$ the expression of the performance measures change. This particular case is explained in Appendix A.

The first performance measure we derive is the service level.

Proposition 3.2.2. *In Erlang CL systems, if* $s \neq a$ *the service level can be written as*

$$\mathrm{SL} = \frac{1}{1-\pi(N)}\left[1 - \Big(A(s) + \pi(N)\Big)\right]$$

with

$$\begin{aligned} A(s) =& \frac{(\mu\tau)^{N-s+1}}{(N-s-1)!}\frac{a^N}{(s-1)!}\pi(0)e^{-\mu s\tau}\cdot \\ & \cdot \int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}\int_0^{+\infty} e^{-a\mu\tau(1+t)u}(1+u)^{N-s-1}dudt \end{aligned}$$

where the function $G(a,s)$ *is defined as in Theorem 3.1.1.*

Proof. Recall first the expression for the service level in Erlang CL systems found in the previous chapter stated in Proposition 2.2.2:

$$\mathrm{SL} = \frac{1}{1-\pi(N)}\left[1 - \left(\sum_{i=0}^{N-s-1}\pi(s+i)\sum_{k=0}^{i}\frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} + \pi(N)\right)\right].$$

Observe that we just showed by Proposition 3.2.1 that we can rewrite the stationary distribution with a continuous function with respect to s, therefore we can already consider the term $\pi(N)$ in the previous expression as a continuous function of the number of agents. Then, we are interested in proving the following equality

$$A(s) = \sum_{i=0}^{N-s-1}\pi(s+i)\sum_{k=0}^{i}\frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} \qquad (\star)$$

where the function $A(s)$ is defined in the statement of the proposition. We can start with the right-hand side of Equation $(\star)$ by using the definition of the function $G(a,s)$ given in Theorem 3.1.1 by taking as input values $\mu s\tau$ and $i+1$ respectively. Then,

$$\begin{aligned}
\text{RHS} &= \sum_{i=0}^{N-s-1} \pi(s+i) \sum_{k=0}^{i} \frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \sum_{k=0}^{i} \frac{(\mu s\tau)^k}{k!} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \frac{(\mu s\tau)^{i+1}}{(i+1)!} \sum_{k=0}^{i} \frac{(\mu s\tau)^{k-(i+1)}(i+1)!}{k!} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \frac{(\mu s\tau)^{i+1}}{(i+1)!} G(\mu s\tau, i+1) \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \frac{(\mu s\tau)^{i+1}}{(i+1)!} \int_0^{+\infty} (i+1) e^{-\mu s\tau t} (1+t)^i dt.
\end{aligned}$$

Now since the summation is finite we can exchange it with the integral and obtain the following:

$$\begin{aligned}
\text{RHS} &= e^{-\mu s\tau} \int_0^{+\infty} e^{-\mu s\tau t} \sum_{i=0}^{N-s-1} \pi(s+i) \frac{(\mu s\tau)^{i+1}}{(i+1)!} (i+1)(1+t)^i dt \\
&= \mu s\tau e^{-\mu s\tau} \int_0^{+\infty} e^{-\mu s\tau t} \sum_{i=0}^{N-s-1} \pi(s+i) \frac{(\mu s\tau(1+t))^i}{i!} dt.
\end{aligned}$$

We can explicitly write the continuous formula for the stationary distribution $\pi(s+i) = \left(\frac{a}{s}\right)^i \frac{a^s}{s!}\pi(0)$, where we already know that $\pi(0)$ is a continuous function of the staffing level s by Proposition 3.2.1. Therefore,

$$\begin{aligned}
\text{RHS} &= \mu s\tau e^{-\mu s\tau} \int_0^{+\infty} e^{-\mu s\tau t} \sum_{i=0}^{N-s-1} \left(\frac{a}{s}\right)^i \frac{a^s}{s!} \pi(0) \frac{(\mu s\tau(1+t))^i}{i!} dt \\
&= \frac{a^s}{s!} \pi(0) \mu s\tau e^{-\mu s\tau} \int_0^{+\infty} e^{-\mu s\tau t} \sum_{i=0}^{N-s-1} \frac{(a\mu\tau(1+t))^i}{i!} dt.
\end{aligned}$$

By considering again the definition of the function $G(a,s)$ of Theorem 3.1.1 with

input values $a\mu\tau(1+t)$ and $N-s$ respectively, we can rewrite

$$\begin{aligned}
\text{RHS} =& \frac{a^s}{s!}\pi(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\frac{(a\mu\tau(1+t))^{N-s}}{(N-s)!}G(a\mu\tau(1+t),N-s)dt \\
=& \frac{(a\mu\tau)^{N-s}}{(N-s)!}\frac{a^s}{s!}\pi(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}G(a\mu\tau(1+t),N-s)dt \\
=& \frac{(\mu\tau)^{N-s+1}}{(N-s)!}\frac{a^N}{(s-1)!}\pi(0)e^{-\mu s\tau}\cdot \\
& \cdot\int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}\int_0^{+\infty}(N-s)e^{-a\mu\tau(1+t)u}(1+u)^{N-s-1}dudt \\
=& \frac{(\mu\tau)^{N-s+1}}{(N-s-1)!}\frac{a^N}{(s-1)!}\pi(0)e^{-\mu s\tau}\cdot \\
& \cdot\int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}\int_0^{+\infty} e^{-a\mu\tau(1+t)u}(1+u)^{N-s-1}dudt
\end{aligned}$$

which proves the equality of the Equation $(\star)$. □

By this proposition, we can indeed write the service level of Erlang CL systems as a composition of continuous equations with respect to the staffing level s. The next performance measure we want to rewrite as a continuous function of s is the average waiting time.

Proposition 3.2.3. *In Erlang CL systems, if $s \neq a$ the average speed of answer can be written as follows:*

$$\text{ASA} = \frac{\frac{a^s}{s!}\pi(0)}{1-\pi(N)}\cdot\frac{1}{\mu(s-a)}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right];$$

where $\pi(0)$ and $\pi(N)$ can be written as continuous functions of s as in Proposition 3.2.1.

Proof. Recall first the expression of ASA given in the previous chapter in Proposition 2.2.3

$$\text{ASA} = \frac{1}{1-\pi(N)}\sum_{i=0}^{N-s-1}\frac{i+1}{s\mu}\pi(s+i).$$

Consider the stationary distribution previously found $\pi(s+i) = \left(\frac{a}{s}\right)^i\frac{a^s}{s!}\pi(0)$, where $\pi(0)$ is a continuous function of s. Then,

$$\begin{aligned}
\text{ASA} &= \frac{1}{1-\pi(N)}\sum_{i=0}^{N-s-1}\frac{i+1}{s\mu}\left(\frac{a}{s}\right)^i\frac{a^s}{s!}\pi(0) \\
&= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s}\sum_{i=0}^{N-s-1}(i+1)\left(\frac{a}{s}\right)^i.
\end{aligned}$$

Consider now only the summation term and develop the calculation as follows by substituting $j = i + 1$:

$$\sum_{i=0}^{N-s-1}(i+1)\left(\frac{a}{s}\right)^i = \frac{s}{a}\sum_{i=0}^{N-s-1}(i+1)\left(\frac{a}{s}\right)^{i+1} = \frac{s}{a}\sum_{j=1}^{N-s} j\left(\frac{a}{s}\right)^j.$$

By the known result of the finite sum $\sum_{j=1}^{n} jx^j = x\frac{1-x^n}{(1-x)^2} - \frac{nx^{n+1}}{1-x}$ for $x \neq 1$, we can rewrite the previous sum as follows since $a \neq s$:

$$\begin{aligned}
\frac{s}{a}\sum_{j=1}^{N-s} j\left(\frac{a}{s}\right)^j &= \frac{s}{a}\left[\frac{a}{s}\cdot\frac{1-\left(\frac{a}{s}\right)^{N-s}}{(1-\frac{a}{s})^2} - \frac{(N-s)\left(\frac{a}{s}\right)^{N-s+1}}{1-\frac{a}{s}}\right] \\
&= \frac{s}{a}\cdot\frac{\frac{a}{s}}{1-\frac{a}{s}}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right] \\
&= \frac{1}{1-\frac{a}{s}}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right].
\end{aligned}$$

Given this result we can substitute in the last expression of ASA we had:

$$\begin{aligned}
\text{ASA} &= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s}\sum_{i=0}^{N-s-1}(i+1)\left(\frac{a}{s}\right)^i \\
&= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s}\cdot\frac{1}{1-\frac{a}{s}}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right] \\
&= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))}\cdot\frac{1}{\mu(s-a)}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right]
\end{aligned}$$

which is what we wanted to prove when $a \neq s$. □

The last performance measure we want to consider is the occupancy of a single server in the Erlang CL model. Recall that the occupancy can be computed by

$$\text{OCC} = \frac{\lambda}{s\mu}(1-\pi(N))$$

which is already a composition of continuous functions with respect to s. For this reason, no extra computations are needed for the occupancy of a single server.

3.2.3 Limiting distribution and performance measures

Based on the structure of the two models explained in the previous chapter, it is possible to prove that for $N \to +\infty$ the performance measures of Erlang CL models are equal to the ones for the Erlang C. In particular, we can prove first the equivalence between the stationary distributions of the two models.

Proposition 3.2.4. *Given the stationary distribution π_{CL} of Erlang CL model in Proposition 3.2.1, for $a < s$ the following limit holds:*

$$\lim_{N \to +\infty} \pi_{CL} = \pi_C$$

where π_C is the stationary distribution of Erlang C model stated in Proposition 3.1.2.

Proof. Recall first that the state space of the Erlang CL model is $S = \{0, 1, \dots, N\}$, and we want to prove the limit stated in the proposition for both cases $x < s$ and $s \leq x \leq N$, where s is the staffing level.

Consider first π_{CL} for $x < s$:

$$\pi_{CL}(x) = \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right)\right]^{-1}$$

where we know from Theorem 3.1.1 that the function $G(a, s)$ does not depend on N. If $a < s$ we have that for $x < s$

$$\begin{aligned}
\lim_{N \to +\infty} \pi_{CL}(x) &= \lim_{N \to +\infty} \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right)\right]^{-1} \\
&= \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{1}{1 - \frac{a}{s}}\right)\right]^{-1} \\
&= \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{s}{s - a}\right)\right]^{-1} \\
&= \pi_C(x)
\end{aligned}$$

which is the stationary distribution for $x < s$ of Erlang C stated in Proposition 3.1.2, which indeed holds for $a < s$.

Consider now $s \leq x \leq N$, then we have

$$\pi_{CL}(x) = \left(\frac{a}{s}\right)^{x-s}\left[G(a,s) + \frac{1 - \left(\frac{a}{s}\right)^{N+1-s}}{1 - \frac{a}{s}}\right]^{-1}$$

from Proposition 3.2.1. Then, again for $a < s$, we have the following limit:

$$\begin{aligned}\lim_{N\to+\infty} \pi_{CL}(x) &= \lim_{N\to+\infty} \left(\frac{a}{s}\right)^{x-s} \left[G(a,s) + \frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right]^{-1} \\ &= \left(\frac{a}{s}\right)^{x-s} \left[G(a,s) + \frac{1}{1-\frac{a}{s}}\right]^{-1} \\ &= \pi_C(x)\end{aligned}$$

which is the stationary distribution for Erlang C models given by Proposition 3.1.2 and this concludes the proof. □

Proposition 3.2.5. *Given the service level* $\mathrm{SL_{CL}}$ *of Erlang CL model in Proposition 3.2.2, for* $a < s$ *the following limit holds:*

$$\lim_{N\to+\infty} \mathrm{SL_{CL}} = \mathrm{SL_C}$$

where $\mathrm{SL_C}$ *is the service level of Erlang C model stated in Proposition 3.2.2.*

Proof. Due to the difficult expression of the service level for Erlang CL models, we need to make some previous calculations before computing the limit. Recall first that

$$\mathrm{SL_{CL}} = \frac{1}{1-\pi(N)}\left[1 - \Big(A(s) + \pi(N)\Big)\right]$$

with

$$\begin{aligned}A(s) =& \frac{(\mu\tau)^{N-s+1}}{(N-s-1)!}\frac{a^N}{(s-1)!}\pi_{CL}(0)e^{-\mu s\tau}\cdot \\ &\cdot \int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s} \int_0^{+\infty} e^{-a\mu\tau(1+t)u}(1+u)^{N-s-1}dudt,\end{aligned}$$

while

$$\mathrm{SL_C} = 1 - \frac{e^{-(s\mu-\lambda)\tau}}{1+G(a,s)\cdot(1-\frac{a}{s})}$$

where $G(a,s)$ is defined as in Theorem 3.1.1, as stated in Proposition 3.1.4. Therefore, in order to prove the limit stated in this proposition we may also just prove that

$$\lim_{N\to+\infty} A(s) = \frac{e^{-(s\mu-\lambda)\tau}}{1+G(a,s)\cdot(1-\frac{a}{s})}$$

because $\pi_{CL}(N) \to 0$ for $N \to +\infty$, as stated in Proposition 3.2.4. Notice also that $\pi_{CL}(0) \to \pi_C(0)$.

Recall the proof of Proposition 3.2.2 where the following equality was proven

$$A(s) = \frac{a^s}{s!}\pi_{CL}(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\sum_{i=0}^{N-s-1}\frac{(a\mu\tau(1+t))^i}{i!}dt.$$

Define now the following function

$$H(r) = \sum_{i=0}^{r-1}\frac{x^i}{i!} = \frac{x^r}{\Gamma(r)}\int_0^{+\infty} e^{-xt}(1+t)^{r-1}dt$$

where we know that it admits an extension for all positive real values of r due to the existence of the function G defined in Theorem 3.1.1. We need to prove that the function $H(r)$ is an increasing monotone function and that

$$\lim_{r\to+\infty} H(r) = e^x.$$

We start by proving the limit. Consider the following computations by using the change of variables $u = x(1+t)$:

$$\begin{aligned}
H(r) &= \frac{x^r}{\Gamma(r)}\int_0^{+\infty} e^{-xt}(1+t)^{r-1}dt \\
&= \frac{e^x}{\Gamma(r)}\int_0^{+\infty} e^{-x(1+t)}[x(1+t)]^{r-1}xdt \\
&= \frac{e^x}{\Gamma(r)}\int_x^{+\infty} e^{-u}u^{r-1}du \\
&= e^x\frac{\int_x^{+\infty} e^{-u}u^{r-1}du}{\int_0^{+\infty} e^{-u}u^{r-1}du}
\end{aligned}$$

where we explicitly write the expression of the Gamma function. Notice now that we can write

$$\int_x^{+\infty} e^{-u}u^{r-1}du = \int_0^{+\infty} e^{-u}u^{r-1}du - \int_0^x e^{-u}u^{r-1}du$$

then,

$$\begin{aligned}
H(r) &= e^x\frac{\int_x^{+\infty} e^{-u}u^{r-1}du}{\int_0^{+\infty} e^{-u}u^{r-1}du} \\
&= e^x\frac{\int_0^{+\infty} e^{-u}u^{r-1}du - \int_0^x e^{-u}u^{r-1}du}{\int_0^{+\infty} e^{-u}u^{r-1}du} \\
&= e^x - e^x\frac{\int_0^x e^{-u}u^{r-1}du}{\int_0^{+\infty} e^{-u}u^{r-1}du}.
\end{aligned}$$

Consider now the following inequality

$$\left|\int_0^x e^{-u}u^{r-1}du\right| \leq \int_0^x x^{r-1}du = \frac{x^r}{r}$$

hence,

$$|H(r) - e^x| \leq e^x \frac{x^r}{r\Gamma(r)} = e^x \frac{x^r}{\Gamma(r+1)} \xrightarrow{r\to+\infty} 0$$

which implies that $H(r) \to e^x$ as $r \to +\infty$.

After proving the limit of the function $H(r)$, we want to prove that it is an increasing monotone function. For our purposes, it is enough to prove that $H(r) \leq H(r+1)$ because in our case we are taking $\lim_{N\to+\infty} H(N+s-1)$. This inequality follows from the fact that we can rewrite $H(r+1) = H(r) + \frac{x^r}{\Gamma(r+1)}$. Indeed, by integrating per parts we have

$$\begin{aligned}
H(r+1) &= e^x \frac{\int_x^{+\infty} e^{-u}u^r du}{\Gamma(r+1)} \\
&= \frac{e^x}{r\Gamma(r)} \int_x^{+\infty} \frac{d}{du}(-e^{-u})u^r du \\
&= \frac{e^x}{r\Gamma(r)} \left[e^{-x}x^r + \int_x^{+\infty} e^{-u}ru^{r-1}du\right] \\
&= \frac{x^r}{r\Gamma(r)} + \frac{e^x}{\Gamma(r)} \int_x^{+\infty} e^{-u}u^{r-1}du \\
&= \frac{x^r}{\Gamma(r+1)} + H(r).
\end{aligned}$$

We can now conclude that also the function $\sum_{i=0}^{N-s-1} \frac{(a\mu\tau(1+t))^i}{i!}$ is an increasing monotone function that for $N \to +\infty$ tends to $e^{a\mu\tau(1+t)}$. Therefore by Monotone Convergence Theorem, we can exchange the limit and the integral and we obtain

the following:

$$\begin{aligned}
\lim_{N\to+\infty} A(s) &= \lim_{N\to+\infty} \frac{a^s}{s!}\pi_{CL}(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\sum_{i=0}^{N-s-1}\frac{(a\mu\tau(1+t))^i}{i!}dt \\
&= \frac{a^s}{s!}\pi_C(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\lim_{N\to+\infty}\sum_{i=0}^{N-s-1}\frac{(a\mu\tau(1+t))^i}{i!}dt \\
&= \frac{a^s}{s!}\pi_C(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}e^{a\mu\tau(1+t)}dt \\
&= \frac{a^s}{s!}\pi_C(0)\mu s\tau e^{-\mu\tau(s-a)}\int_0^{+\infty} e^{-\mu\tau t(s-a)}dt \\
&= \frac{a^s}{s!}\pi_C(0)\mu s\tau e^{-\mu\tau(s-a)}\frac{1}{-\mu\tau(s-a)}e^{-\mu\tau t(s-a)}\Big|_0^{+\infty} \\
&= \frac{a^s}{s!}\pi_C(0)\mu s\tau e^{-\mu\tau(s-a)}\int_0^{+\infty} e^{-\mu\tau t(s-a)}dt \\
&= \frac{a^s}{s!}\pi_C(0)\mu s\tau e^{-\tau(\mu s-\lambda)}\frac{1}{\mu\tau(s-a)} \\
&= \frac{a^s}{s!}\pi_C(0)e^{-\tau(\mu s-\lambda)}\frac{s}{s-a} \\
&= \frac{a^s}{s!}\frac{1}{\frac{a^s}{s!}\left(G(a,s)+\frac{s}{s-a}\right)}\frac{s}{s-a}e^{-\tau(\mu s-\lambda)} \\
&= \frac{a^s}{s!}\frac{1}{\frac{a^s}{s!}\left(G(a,s)+\frac{1}{1-a/s}\right)}\frac{1}{1-\frac{a}{s}}e^{-\tau(\mu s-\lambda)} \\
&= \frac{1}{\left(1-\frac{a}{s}\right)G(a,s)+1}e^{-\tau(\mu s-\lambda)}
\end{aligned}$$

which is the continuous formula for the service level of Erlang C models.

□

Proposition 3.2.6. *Given the average speed of answer* $\mathrm{ASA_{CL}}$ *of Erlang CL model in Proposition 3.2.3, for* $a < s$ *the following limit holds:*

$$\lim_{N\to+\infty}\mathrm{ASA_{CL}} = \mathrm{ASA_C}$$

where $\mathrm{ASA_C}$ *is the average speed of answer in Erlang C models stated in Proposition 3.1.5.*

Proof. Recall first the expression of $\mathrm{ASA_{CL}}$ of Proposition 3.2.3:

$$\mathrm{ASA_{CL}} = \frac{\frac{a^s}{s!}\pi_{CL}(0)}{1-\pi_{CL}(N)}\cdot\frac{1}{\mu(s-a)}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right]$$

where we know that the function $G(a,s)$ does not depend on N. Notice also that for Proposition 3.2.4, we have the following

$$\pi_{CL}(0) \longrightarrow \pi_C(0)$$
$$\pi_{CL}(N) = \left(\frac{a}{s}\right)^{N-s} \frac{a^s}{s!}\pi_{CL}(0) \longrightarrow 0$$

for $N \to +\infty$, because $a < s$.

Therefore, we can compute the limit for $a < s$ as follows:

$$\begin{aligned}
\lim_{N\to+\infty} \mathrm{ASA_{CL}} &= \lim_{N\to+\infty} \frac{\frac{a^s}{s!}\pi_{CL}(0)}{1-\pi_{CL}(N)} \cdot \frac{1}{\mu(s-a)}\left[\frac{1-\left(\frac{a}{s}\right)^{N-s}}{1-\frac{a}{s}} - (N-s)\left(\frac{a}{s}\right)^{N-s}\right] \\
&= \frac{a^s}{s!}\pi_C(0) \cdot \frac{1}{\mu(s-a)} \cdot \frac{1}{1-\frac{a}{s}} \\
&= \frac{a^s}{s!}\left[\frac{a^s}{s!}\left(G(a,s) + \frac{s}{s-a}\right)\right]^{-1} \cdot \frac{1}{\mu(s-a)} \cdot \frac{1}{1-\frac{a}{s}} \\
&= \frac{1}{G(a,s) + \frac{1}{1-\frac{a}{s}}} \cdot \frac{1}{1-\frac{a}{s}} \cdot \frac{1}{\mu(s-a)} \\
&= \frac{1}{(1-\frac{a}{s})G(a,s)+1} \cdot \frac{1}{s\mu(1-\frac{a}{s})} \\
&= \frac{1}{s\mu(1-\frac{a}{s})((1-\frac{a}{s})G(a,s)+1)} \\
&= \mathrm{ASA_C}
\end{aligned}$$

which is the average waiting time for Erlang C models given in Proposition 3.1.5. Notice that in the previous computations we used the fact that for $N \to +\infty$ we have $(N-s)\left(\frac{a}{s}\right)^{N-s} \to 0$ because the exponential term goes to zero more quickly than $N-s$ to infinity. □

Proposition 3.2.7. *Given the occupancy* $\mathrm{OCC_{CL}}$ *of Erlang CL models, for* $a < s$ *the following limit holds:*

$$\lim_{N\to+\infty} \mathrm{OCC_{CL}} = \mathrm{OCC_C}$$

where $\mathrm{OCC_C}$ *is the occupancy of a single server in Erlang C models.*

Proof. Recall that

$$\mathrm{OCC_{CL}} = \frac{\lambda}{s\mu}\big(1-\pi_{CL}(N)\big)$$

is the occupancy of a single server in Erlang CL systems. We can write explicitly the blocking probability as $\pi_{CL}(N) = \left(\frac{a}{s}\right)^{N-s}\frac{a^s}{s!}\pi_{CL}(0)$, then for $a < s$ the blocking

probability goes to zero for $N \to \infty$, because by Proposition 3.2.4 we have that $\pi_{CL}(0) \to \pi_C(0)$. Therefore,

$$\lim_{N \to +\infty} \mathrm{OCC_{CL}} = \lim_{N \to +\infty} \frac{\lambda}{s\mu}\big(1 - \pi_{CL}(N)\big) = \frac{\lambda}{s\mu} = \mathrm{OCC_C}$$

which is indeed the occupancy of a single server in Erlang C models and it concludes the proof. □

Chapter 4

Numerical implementation of Erlang C model

After introducing and describing the continuous models for Erlang systems in the previous chapter, the focus will be on the implementation of these models. The programming language used at CCmath for the implementation of the Erlang models is C++. The biggest challenge of this part of the project is finding a suitable method to implement the integrals used to define the new version of the performance measures stated in the previous chapter. This programming language does not provide any particular functions to directly compute the actual values of these integrals. However, there are various methods of numerical integration that allow us to calculate them with different degrees of precision. For this reason, after introducing these methods we will make a short analysis via simulation in order to determine the optimal method for our implementation.

In order to verify the precision of the results, the model has been also implemented in Python, where due to some specific functions it is possible to compute the exact value of integrals. For this reason, it is possible to obtain the actual value of performance measures. This would be useful for the analysis of performance of the implementation in C++.

Another challenge presented in this chapter is the computation of the fractional number of agents required to satisfy a certain target of a performance measure. The staffing level can be obtained via direct computation or via methods of convergence, which will be explained later. In addition, a short analysis of performance in terms of time execution will be presented in order to verify which is the most suitable convergence method for the performance measures of Erlang C systems.

In this chapter, the implementation of Erlang C continuous formulas is presented. We show first how to obtain the performance measures when the fractional number of agents is given and then we will focus on how to obtain the fractional staffing level that allows obtaining a certain target for a performance measure.

4.1 Numerical integration methods

In this section, we introduce different numerical integration methods that have been used in this project in order to compute integrals in C++. Later in this chapter, it will be explained which method has been chosen for the Erlang C model, according to a simulation to test the performance of all the following methods.

The numerical integration models we present are the *Rectangle rule*, the *Trapezoidal rule* and the *Simpson rule*.

Theorem 4.1.1 (Rectangle rule)**.** *Consider the interval* $[a, b]$*, an integrable function* $f(t)$ *for all* $t \in [a, b]$ *and a number of steps* $n \in \mathbb{N}$*. Then, the integral of* $f(t)$ *over the interval* $[a, b]$ *can be approximated as follows:*

$$\int_a^b f(t)dt \approx \sum_{i=0}^{n-1} f\left(\frac{t_i + t_{i+1}}{2}\right)(t_{i+1} - t_i)$$

where the set of intervals $\{[t_i, t_{i+1})\}_{i=0}^{n}$ *is a partition of* $[a, b]$ *such that* $a = t_0 < t_1 < \cdots < t_n = b$*.*

Intuitively, Theorem 4.1.1 presents an approximation for an integral as the sum of infinitesimal rectangles whose length is the length of a single interval and as height of each rectangle we consider the value of the integrand function evaluated in the middle point of that interval as presented in Figure 4.1.1.

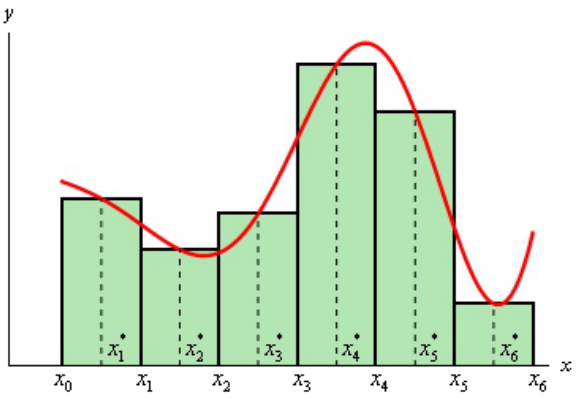

Figure 4.1.1: Rectangle rule for numerical integration.

Notice that the Rectangle rule can be also applied by taking as height of each rectangle the evaluation of the integrand function in one of the extremes of the interval, not only in the middle point.

The second method for numerical integration is stated in the following theorem.

Theorem 4.1.2 (Trapezoidal rule)**.** *Consider the interval* $[a, b]$*, an integrable function* $f(t)$ *for all* $t \in [a, b]$ *and a number of steps* $n \in \mathbb{N}$*. Then, the integral of* $f(t)$ *over the interval* $[a, b]$ *can be approximated as follows:*

$$\int_a^b f(t)dt \approx \sum_{i=0}^{n-1} \left(\frac{f(t_i) + f(t_{i+1})}{2} \right) (t_{i+1} - t_i)$$

where the set of intervals $\{[t_i, t_{i+1})\}_{i=0}^n$ *is a partition of* $[a, b]$ *such that* $a = t_0 < t_1 < \cdots < t_n = b$*.*

The idea behind Theorem 4.1.2 is the same as Theorem 4.1.1, however, instead of summing rectangles we are considering trapezoids where each one of them is defined by the segment between the evaluations of the extremes of each integral through the integrand function f, as shown in Figure 4.1.2.

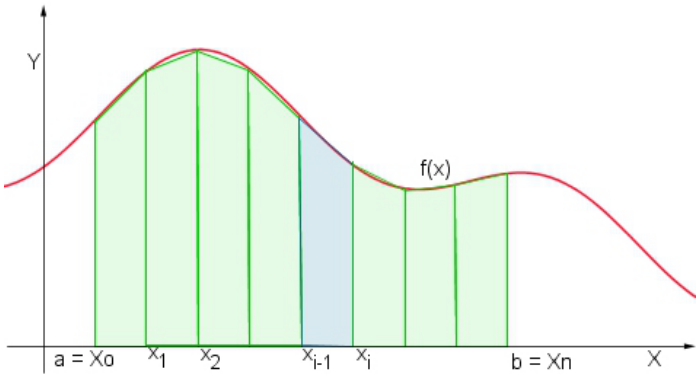

Figure 4.1.2: Trapezoidal rule for numerical integration.

The last numerical integration method used for this project is stated in the following theorem:

Theorem 4.1.3 (Simpson rule)**.** *Consider the interval* $[a, b]$*, an integrable function* $f(t)$ *for all* $t \in [a, b]$ *and a number of steps* $n \in \mathbb{N}$*. Then, the integral of* $f(t)$ *over the interval* $[a, b]$ *can be approximated as follows:*

$$\int_a^b f(t)dt \approx \sum_{i=0}^{n-1} \left(\frac{f(t_i) + 4f\left(\frac{t_i + t_{i+1}}{2}\right) + f(t_{i+1})}{6} \right) (t_{i+1} - t_i)$$

where the set of intervals $\{[t_i, t_{i+1})\}_{i=0}^n$ *is a partition of* $[a, b]$ *such that* $a = t_0 < t_1 < \cdots < t_n = b$*.*

As stated in Theorem 4.1.3, the approximation of the integral in each interval is made by using a polynomial passing through the function f evaluated in the

extremes of the interval and the middle point of the interval. Figure 4.1.3 shows a simple example of the Simpson rule applied on a single interval $[x_0, x_2]$, where the black line represents the integrand function f and the highlighted area is the approximated integral computed by using Simpson rule.

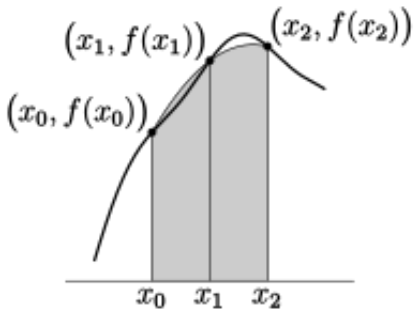

Figure 4.1.3: Simpson rule for numerical integration in a single interval.

Notice first that the Rectangle rule requires only one evaluation of the function f, while the Trapezoidal rule two and the Simpson rule three. This fact might be relevant when the integrand function is particularly expensive in terms of computation. On the other hand, it is important to analyze the speed of convergence, since one method might need a lower number of steps to reach a good approximation of the integral.

In order to do that, consider a simple example where we want to integrate a function whose primitive cannot be easily computed, as $f(x) = 19e^{-15x}(1+x)^{18}$. This is the function we want to integrate with these methods to compute the function $G(a, s)$ when $s = 19$ and $a = 15$.

Notice that the exact value of this integral is 14.699816. Table 4.1.1 shows the result of the integral obtained by using respectively the Rectangle rule, the Trapezoidal rule and the Simpson rule, for different values of n.

As it is presented in Table 4.1.1, we can notice that for this example the method that reaches the exact solution with the lowest number of steps is the Simpson method, while the other methods require higher values of n. For the Simpson method the error between the actual value of the integral and the approximation is null when the number of steps is around 500, while for both the rectangle and the trapezoidal rule the precision is reached when $n = 10000$. For this reason, we can consider the Simpson rule as the least expensive to integrate this type of function.

Speed of convergence for different n			
n	Rectangle	Trapezoidal	Simpson
5	1.197458	-2.100591	0.098109
10	0.235089	-0.451566	0.006204
50	0.00858	-0.017134	9e-06
100	0.002139	-0.004277	1e-06
500	8.6e-05	-0.000171	0.0
1000	2.1e-05	-4.3e-05	0.0
5000	1e-06	-2e-06	0.0
10000	0.0	0.0	0.0

Table 4.1.1: Comparison of the speed of convergence of the three different methods for $n \in \{5, 10, 50, 100, 500, 1000, 5000, 10000\}$ in a simple example. In particular, this table shows the error between the exact computation of the integral and the approximation obtained with the numerical integration methods.

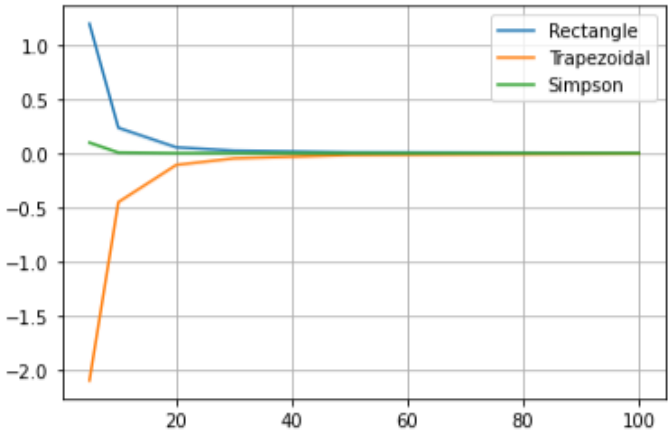

Figure 4.1.4: Error of numerical integration methods for $n \leq 100$.

In Figure 4.1.4, it is possible to see how fast the error function of every numerical integration method goes to zero for increasing values of n. It is visible how even for very low values of n the Simpson method is closer to the exact solution with respect to the other two methods.

4.2 Convergence methods

In this section, we present the methods of convergence considered for this research. In particular, we will explain the importance of these methods for the computation of the fractional staffing level s, when a certain target T of a performance measure is given.

These methods are based on the research of zeros of a certain function. In our case, we want to find the zero of the following

$$f(s) = T - v(s)$$

where T is the target we want to achieve and $v(s)$ is a function that describes the performance measure computed through the fractional staffing level s. By following this idea, it is possible to select $s^\star \in \mathbb{R}_+$ such that $f(s^\star) = 0$, i.e., $v(s^\star) = T$. The scalar $s^\star$ is indeed called zero of the function f.

These methods belong to the class of iterative methods: for a given initial value s_0, these methods generate a sequence $\{s_k\}_{k\in\mathbb{N}}$ such that $s_k \to s^\star$ for $k \to +\infty$. This convergence might be *local* or *global*, depending on the choice of the initial value s_0. For our purpose, it is sufficient to consider local convergence, since later in this chapter we will show how to choose the value s_0 in a certain neighborhood of $s^\star$.

We will discuss four different methods of convergence: all of them converge to the exact solution, however, it is important to decide which one is preferable by analyzing the speed of convergence and the computational cost. The speed of convergence indicates how rapidly the values s_k obtained at every step get closer and closer to the zero $s^\star$ and it is defined by the order of convergence. On the other hand, the computational cost is quantified by the required number of evaluations of the function for each iteration. This is a crucial part of the analysis of performance since the evaluation of the function f might be computationally expensive.

Furthermore, it is essential to introduce a stop criterion because the process generated by an iterative method does not end, therefore we need to interrupt the process at a certain step k when s_k is sufficiently close to the zero $s^\star$. In this project, we decided to consider a certain tolerance $\text{TOL} = 1E-8$ and we run the algorithm as long as the norm of the function f is greater than this tolerance.

Suitable methods of convergence for this problem are the *Newton method* and its variations, which all have the following general form:

$$\begin{cases} \text{Given } x_0 \in I(x^\star) \\ \text{For } k = 0, 1, 2, \dots \\ x_{k+1} = x_k - b_k^{-1} \cdot f(x_k) \\ \text{STOP if } ||f(x_k)|| \leq \text{TOL} \end{cases} \tag{4.2.1}$$

Notice that it is essential to define some general hypothesis in order to guarantee convergence for every different method. These hypotheses hold for all different methods used in this research and state important properties for finding a solution of Equation 4.2.1.

Let $\mathcal{D} \subset \mathbb{R}$ be an open interval and $f : \mathcal{D} \to \mathbb{R}$. Suppose that:

- $H1$: f is continuously differentiable in $\mathcal{D}$;
- $H2$: Equation 4.2.1 has solution $x^\star \in \mathcal{D}$;
- $H3$: $f' \in Lip_L(\mathcal{D})$ and there exists $\rho > 0$ such that $|f'(x)| \geq \rho$ for all $x \in \mathcal{D}$.

Notice that the set $Lip_L(\mathcal{D})$ is the set of all Lipschitz functions with constant L on the set $\mathcal{D}$, i.e., the set of all functions $F : \mathcal{D} \to \mathbb{R}$ such that

$$||F(y) - F(x)|| \leq L||y - x||$$

for all $x, y \in \mathcal{D}$, where $||\cdot||$ is a norm defined on $\mathbb{R}$.

The convergence methods we decided to consider for this research are the *Newton method*, *stationary Newton method*, the *Newton method by differences* and the *secant method*.

Newton method

The Newton method satisfies Equation 4.2.1, where at each step $k = 1, 2, \ldots$ we have the following parameter

$$b_k = f'(x_k).$$

Notice that this method is quite expensive because not only the function f but also its derivative needs to be evaluated at every step k. For this reason, the computational cost of the algorithm is quite large.

Algorithm 4.2.1 Newton method

Given: x_0
For $k = 0, 1, 2, \ldots$
$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$

The good aspect of this method is the speed of convergence, which is quadratic. The following result states the speed of convergence of the Newton method:

Theorem 4.2.1. *Suppose that hypothesis $H1$, $H2$ and $H3$ hold. Then there exists $\eta > 0$ such that $|x_0 - x^\star| < \eta$, then the sequence $\{x_k\}_{k \geq 0}$ generated by Algorithm 4.2.1 is well defined and converges to $x^\star$. Then*

$$|x_{k+1} - x^\star| \leq \frac{L}{2\rho}|x_k - x^\star|^2$$

for all $k = 0, 1, 2, \ldots$, i.e., the Newton method has quadratic convergence.

Compared to many other methods, the Newton method is quite fast. For this reason, we would like to preserve this important quality of the method, while we modify it in order to make it less expensive in terms of computations. Another drawback of this method is the fact that the derivative of the function f might not be defined everywhere, or might be zero. Indeed, Algorithm 4.2.1 is well defined only if $f'(x_k) \neq 0$ for all $k = 0, 1, 2, \ldots$. Therefore, the large field of numerical analysis suggests other methods that follow the same principles of the Newton method and solve these problems at the same time. The main idea to solve this problem is to find a substitute for the computation of the derivative at every step k.

Stationary Newton method

In order to avoid the expensive computation of the derivative at every step k, the stationary Newton method suggests computing the derivative only once and keeping the parameter b_k constant as follows:

$$b_k = f'(x_0).$$

For this method, the derivative of the function f is evaluated only once in x_0.

Algorithm 4.2.2 Stationary Newton method

Given: x_0
For $k = 0, 1, 2, \ldots$
$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}$

Notice that Algorithm 4.2.2 is well defined if and only if $f'(x_0) \neq 0$. In this case, the speed of convergence is stated in the following theorem:

Theorem 4.2.2. *Suppose that hypothesis $H1$, $H2$ and $H3$ hold. Then there exists $\eta > 0$ such that $|x_0 - x^\star| < \eta$, then the sequence $\{x_k\}_{k\geq 0}$ generated by Algorithm 4.2.2 of stationary Newton method converges linearly to $x^\star$.*

Newton method by differences

The Newton method by differences is made by approximating the values of f' by using the definition of the derivative. The idea is to define the parameter b_k of Equation 4.2.1 as

$$b_k = \frac{f(x_k + h) - f(x_k)}{h}$$

Algorithm 4.2.3 Newton method by differences

Given: x_0, h
For $k = 0, 1, 2, \ldots$
$b_k = \frac{f(x_k+h)-f(x_k)}{h}$
$x_{k+1} = x_k - \frac{f(x_k)}{b_k}$

for a small constant h. Therefore the algorithm is the following:

This method requires two evaluations of the function f at each step k: in x_k and in $x_k + h$. The speed of convergence is super-linear as stated in the following theorem.

Theorem 4.2.3. *Suppose that hypothesis $H1$, $H2$ and $H3$ hold. Then there exists $\eta > 0$ such that $|x_0 - x^\star| < \eta$, then the sequence $\{x_k\}_{k\geq 0}$ generated by Algorithm 4.2.3 of Newton method by differences converges super-linearly to $x^\star$.*

Secant method

The secant method is quite useful when the derivative of the function f is unknown. The main idea is that since it is not possible to consider the tangent line on each point $(x_k, f(x_k))$, we want to consider the secant line from $(x_{k-1}, f(x_{k-1}))$ and $(x_k, f(x_k))$. Following this argument, the computation of the derivative is not required and the only function that needs to be evaluated is the function f. Then, the parameter b_k of Equation 4.2.1 is defined as

$$b_k = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

One drawback of this method is that we need to know the value of f for two initial points x_0 and x_{-1} in order to compute the first value of the sequence x_1. Consider

$$b_0 = \frac{f(x_0) - f(x_{-1})}{x_0 - x_{-1}}$$

as the coefficient of the secant line passing through $(x_{-1}, f(x_{-1}))$ and $(x_0, f(x_0))$.

Algorithm 4.2.4 Secant method

Given: x_0, x_{-1}
For $k = 1, 2, \ldots$
$b_k = \frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$
$x_{k+1} = x_k - \frac{f(x_k)}{b_k}$

The algorithm is well defined if and only if $b_k \neq 0$ for all k. Notice also that for every step only one evaluation of the function f in x_k is required since $f(x_{k-1})$ is given from the previous step.

The secant method has a super-linear speed of convergence as stated in the following theorem:

Theorem 4.2.4. *Suppose that hypothesis $H1$, $H2$ and $H3$ hold. Then there exists $\eta > 0$ such that x_0 and x_{-1} are distinct points and are inside the interval with center $x^\star$ and radius η, then the sequence $\{x_k\}_{k\geq 0}$ generated by Algorithm 4.2.4 of secant method converges super-linearly to $x^\star$.*

Summary of convergence methods

In order to compare these methods, we can focus on Table 4.2.1, where a short summary of all their main features is presented.

Summary of convergence methods				
Method	x_0	f	f'	Convergence
Newton	$x_0 \sim x^\star$	2	yes	quadratic
Stat. Newton	$x_0 \sim x^\star$	1	no	linear
Newton by diff.	$x_0 \sim x^\star$	2	no	super-linear
Secant	$x_0 \sim x^\star$	1	no	super-linear

Table 4.2.1: Summary of convergence methods with their main features: convergence from an initial point x_0 to the solution $x^\star$; number of evaluations of the function at each step; evaluation of the derivative of the function at each step; and speed of convergence.

From this table, it is possible to see that if the derivative of the function f exists and the evaluation of f and f' is not too expensive in terms of computations the preferable method is Newton since its convergence is quadratic, i.e., the fastest comparing to all the others. However, if the derivative f' is unknown and the evaluation of the function f is not so expensive, the method we want to select is either Newton by differences or the secant method for their super-linear speed of convergence. If the computation of the function f is expensive, the secant method achieves a good approximation of the solution $x^\star$ with the lowest number of evaluations of f and in a quite small number of iterations.

4.3 Implementation and analysis of performance

As stated in Chapter 3, the performance measure of interest for Erlang C systems is the service level, the average waiting time and the occupancy of a single server. As already mentioned, the occupancy is already a continuous function of the number of agents s, therefore we do not need to find any particular methods to compute it since the implementation is straightforward.

For this reason, in this section we will focus on the implementation of the service level and the average waiting time for the Erlang C systems. In the first part, we will discuss the computation of these performance measures when the number of agents s is given, while in the second part the value of the performance measures will be the target we want to achieve with the fractional number of agents that needs to be computed.

4.3.1 Implementation of performance measures

Recall first the continuous formulas for the service level and the average speed of answer given in Chapter 3 respectively from Proposition 3.1.4 and Proposition 3.1.5:

$$\mathrm{SL} = 1 - \frac{e^{-(s\mu-\lambda)\tau}}{1 + G(a,s)\cdot(1-\frac{a}{s})}$$

$$\mathrm{ASA} = \frac{1}{s\mu(1-\frac{a}{s})(1+G(a,s)\cdot(1-\frac{a}{s}))}$$

where the function $G(a,s)$ stated in Theorem 3.1.1 is the following:

$$G(a,s) = \int_0^\infty s e^{-at}(1+t)^{s-1}\mathrm{d}t.$$

In order to implement this integral function, it is necessary to use a numerical integration method. As previously stated in this chapter, we considered for this research three different rules: Rectangle, Trapezoidal and Simpson.

As already mentioned, with this type of methods for approximation for integrals, the thinner the partition of the interval, the more precise will be the approximation. However, dividing the partition into too many intervals might result to be a downside due to the expensive amount of computations. For this reason, the choice of the number of steps n plays a crucial role in the implementation of the integral.

Analysis of performance of numerical integration

In order to decide which method is the most appropriate for the computation of the function $G(a,s)$, we decided to implement the three rules and we conducted

an analysis of performance via simulation. It is possible to set certain ranges for the input parameters of the Erlang C model. In particular, we decided to consider the following:

- arrival rate $\lambda \in [1, 10]$;
- service rate $\mu \in [0.05, 1]$;
- acceptable waiting time $\tau \in [0.1, 1]$;
- number of agents $s \in (a, 2a]$, where a is the net workload of the system based on the previous parameters.

The simulation has been built by choosing 10 or 11 equidistant values for each parameter in its range previously defined: 10 for λ, 11 for μ, 11 for τ, and 10 for s. We subsequently computed the service level and the average waiting time for all the possible combinations of these values obtaining 12100 results for each numerical integration method. The exact values of the performance measures for all the possible combinations of parameters have been calculated also in Python, by the exact computation of the integral function $G(a, s)$. Given the exact values, it was possible to make a comparison of the result by considering the error as the difference between the exact value and the approximation obtained in C++.

In order to make a good analysis of performance, the errors have been visualized with box-plot diagrams for both the service level and the average speed of answer.

Figure 4.3.4 represents the errors in the computation of the service level for all three numerical integration methods for $n \in \{1000, 10000, 100000\}$. From the box-plot diagrams, we can see how with the Simpson rule already with 10000 steps we almost reach precision, while for the Trapezoidal rule and the Rectangle rule the error does not reach null values even when $n = 100000$. Therefore, we can conclude that for the service level the Simpson rule is the preferred integration method to be used for the computation of the function $G(a, s)$.

On the other hand, Figure 4.3.8 represents the error in the computation of the average waiting time and it is possible to see that the integration method that gets closer to the exact values of the ASA is again the Simpson rule, since for 10000 and 100000 steps the box-plots show that the errors are approximately of the order of 10^{-5}. For the Trapezoidal and Rectangle rules it is visible from the diagrams that the errors are much larger for all values of n analyzed in the simulation.

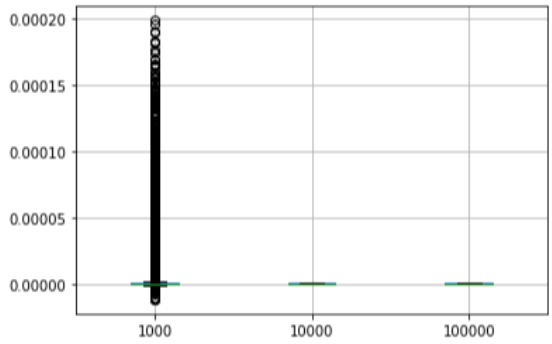

Figure 4.3.1: Simpson rule

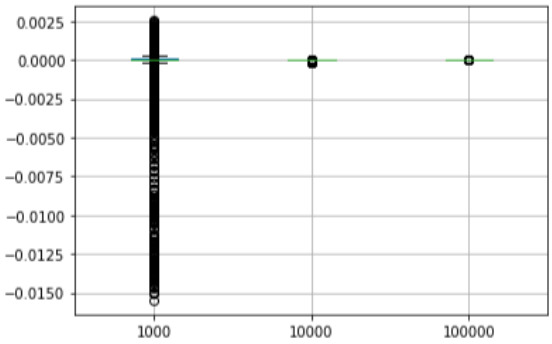

Figure 4.3.2: Trapezoidal rule

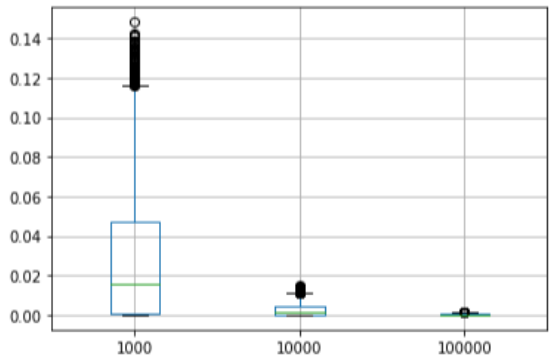

Figure 4.3.3: Rectangle rule

Figure 4.3.4: Box-plot diagrams representing the errors between the simulation results computed by a numerical integration method and the exact values of the continuous computation of the service level for Erlang C models. For each method, the error is presented for different values of n: 1000, 10000 and 100000.

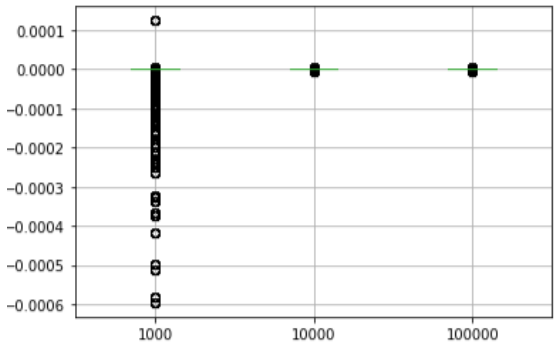

Figure 4.3.5: Simpson rule

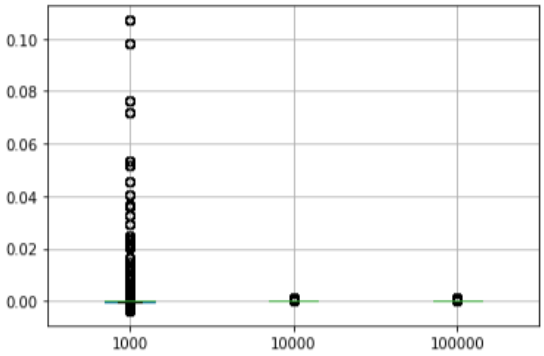

Figure 4.3.6: Trapezoidal rule

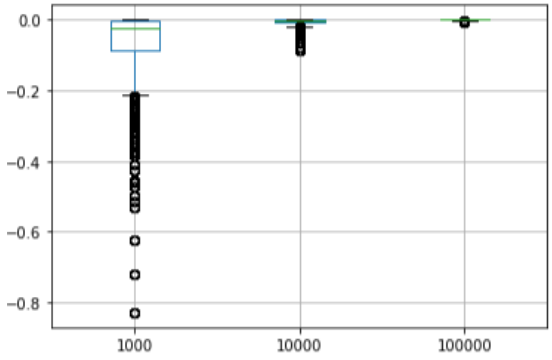

Figure 4.3.7: Rectangle rule

Figure 4.3.8: Box-plot diagrams representing the errors between the simulation results computed by a numerical integration method and the exact values of the continuous computation of the average waiting time for Erlang C models. For each method, the error is presented for different values of n: 1000, 10000 and 100000.

Another important aspect for the analysis of performance of these methods is to make comparisons in terms of the time of execution. Therefore, we summarize in

table 4.3.1 the time execution of the simulation for different numerical integration methods and for different numbers of steps.

Time of execution for different n			
n	Rectangle	Trapezoidal	Simpson
1000	2.423s	2.405s	4.149s
10000	19.172s	18.160s	36.717s
100000	188.949s	176.167s	371.913s

Table 4.3.1: In this table, it is represented the time of execution of the simulation of 12100 different combinations of values in C++ for 1000, 10000 and 100000 steps.

From this table, it is possible to deduce how for larger values of n, the time execution of the simulation increases greatly. Notice also how Rectangle and Trapezoidal rules are less expensive methods in terms of computations compared to the Simpson rule. However, as previously shown, the latter seems to reach good precision already with 10000 steps, while the other two methods require 100000. For this reason, we can conclude that the most preferable method is the Simpson rule with a number of steps $n = 10000$.

It is also important to make a comparison between the different time executions of the methods for a single value of n. Recall that we mentioned previously in this chapter that the Rectangle rule requires only one evaluation of the integrand function, while the Trapezoidal rule requires two and the Simpson rule three. However, this is not reflected in terms of time execution, because as shown in Table 4.3.1 the Rectangle rule and Trapezoidal rule seem to take almost the same amount of time to compute the integral, while the Simpson rule's time is only twice of the Rectangle. The reason why there is this unexpected difference between the three methods lies in the structure of the implementation. Recall that for each interval both the Trapezoidal and the Simpson rules compute the evaluation of the function in the extremes of that interval. In each iteration, the algorithm can reuse one of the evaluations of the extreme points of the previous iteration. This implementation allows the Trapezoidal method to compute only one new evaluation for each step and the Simpson method requires just two evaluations instead of three. For this reason, it appears that the Rectangle and the Trapezoidal methods require the same amount of time to compute the integral, while the time execution of Simpson is double compared to the other numerical integration techniques.

4.3.2 Research of fractional number of agents for a given target

In this section, we analyze the implementation of the continuous Erlang C model, when the input consists of a certain target for the service level, for example, and we are interested in computing the fractional number of agents required in order to meet exactly that target.

In general, we want to understand how to compute the fractional number of agents required to satisfy a certain target for a performance measure.

The first approach for this problem we analyzed in this research consisted on trying to compute the inverse functions of the performance measures with respect to the staffing level s. The goal was to create similar functions as in the theory presented in Chapter 3, by computing the fractional staffing level instead. Eventually, this idea turned out to be too expensive in terms of computations and too complicated due to the integral function $G(a, s)$. Subsequently, the research focused on other ways to solve this problem. The best approach found is using methods of convergence previously described in this chapter. As already mentioned, we want to find the zero of the function $f(s) = T - v(s)$, where T is the target we want to achieve and $v(s)$ is a performance measure computed through the fractional staffing level s. Therefore, the function $v(s)$ might be either the service level or the average waiting time in Erlang C systems.

To begin with, we decided to apply these methods of convergence in certain intervals, chosen accordingly to find a proper initial value x_0 for the iterations of the methods. In particular, given a certain target T that we want to achieve, the idea is to create an interval (s_1, s_2) of length l. Then, we can move the interval along the all possible values of s. By computing the corresponding performance measure v_1 and v_2 for the respective extreme values s_1 and s_2 of every interval, we can find the correct range where the target T falls into.

Notice that it is important to mention that the first interval we consider is $(s_1, s_2) = (a, a + l)$ for which in particular we set $v_1 = 0$. If the target does not fall into (v_1, v_2), we update the value of s_1 as $a + l$ and s_2 as $a + 2l$, and we compute the corresponding values v_1 and v_2. This procedure continues until the target T falls into the interval (v_1, v_2). Once we obtain this, we apply the convergence method for the corresponding interval (s_1, s_2).

Figure 4.3.9 shows how we select the neighborhood for finding the target when the performance measure is the service level. The SL is represented by the blue function and we move the red interval (s_1, s_2) until the target T falls into it, as shown in the figure.

Moreover, when we consider the average speed of answer, we have to take into account that when $s \to a^+$ the value of ASA goes asymptotically to $+\infty$. For this reason, for the first interval (s_1, s_2) we prefer to consider the inverse of the average

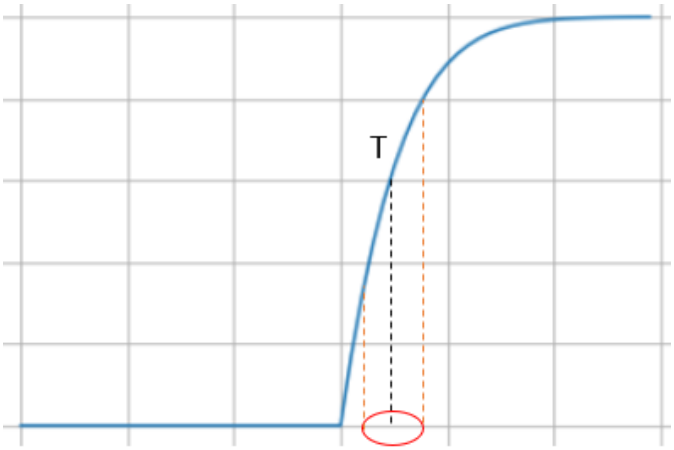

Figure 4.3.9: Service level: research of the neighborhood in which we want to apply a convergence method to find the target T. The interval (s_1, s_2) is represented by the red circle.

waiting time and therefore also the inverse of the target we want to obtain, as shown in Figure 4.3.10. Starting from the interval $(s_1, s_2) = (a + l, a + 2l)$, we consider the regular formula to compute the ASA without taking the inverse, since for larger values of s the direct formula shows a better behavior and it is easier to compute.

By using this approach, it is possible to apply any of the convergence methods by setting $x_0 = s_1$ and computing all the iterations as $s_{k+1} = s_k - f(s_k)/b_k$, even when one of the extremes of the interval coincides with the net workload a.

By following this idea, for both performance measures it is possible to find $s^\star \in \mathbb{R}_+$ such that $f(s^\star) = 0$, i.e., $v(s^\star) = T$. The scalar $s^\star$ is indeed the zero of the function f: in other words it is the fractional number of agents corresponding to the given target T.

It is clear that these methods will have local convergence since we are applying them in a neighborhood of $s^\star$. Therefore, in order to guarantee convergence we need to verify the hypothesis $H1$, $H2$ and $H3$ previously for a general interval (s_1, s_2), where $s_1, s_2 > a$ and the convergence will be guaranteed by the theorems previously stated in this chapter.

Consider first the service level. Therefore, the function we want to consider is

$$f(s) = T - \mathbb{P}(W \leq \tau)$$

by Definition 3.1.4. Then we can check that

- $H1$: f is continuously differentiable because $v(s)$ is also continuously differentiable for all $s > a$ since it is a probability function;

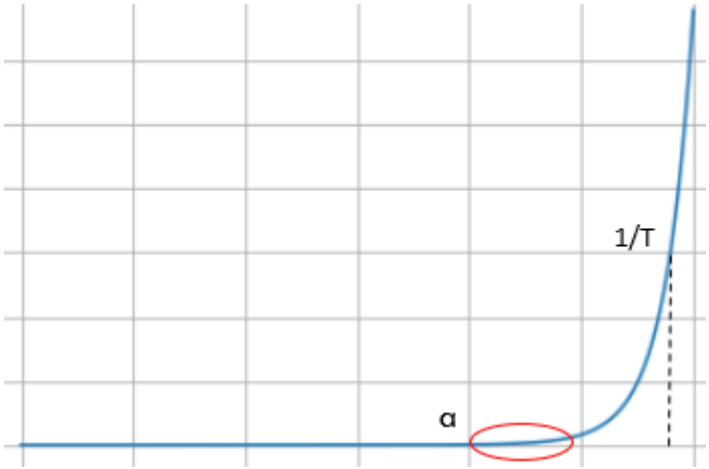

Figure 4.3.10: Inverse of ASA used when $s_1 = a$. Research of the neighborhood in which we want to apply a convergence method to find the target $1/T$. The interval (s_1, s_2) is represented by the red circle.

- $H2$: Equation 4.2.1 has solution for this choice of f;
- $H3$: we need to prove that the derivative f' is a Lipschitz function and that there exists $\rho > 0$ such that $|f'(s)| \geq \rho$ for all $s \in (s_1, s_2)$. Notice first that if $f' \in \mathcal{C}^1$ over an interval A, then f' is Lipschitz on that interval with $L \leq \sup_{s \in A} |f''(s)|$. Notice also that the second requirement of this hypothesis means that $\nexists s \in (s_1, s_2)$ such that $f'(s) = 0$, because then for continuity $\exists \rho > 0$ such that $|f'(s)| > \rho$ for all $s \in (s_1, 2)$. Therefore, we can just check that $|f'(s)| > 0$ for all $s \in (s_1, s_2)$. We can compute first the derivative f' as follows:

$$\begin{aligned} f'(s) &= -v'(s) \\ &= -\frac{d}{ds}\mathbb{P}(W \leq \tau) \\ &= -\frac{d}{ds}\left(1 - \frac{e^{(s\mu-\lambda)\tau}}{1 + G(a,s)(1-\frac{a}{s})}\right) \\ &= e^{(s\mu-\lambda)\tau}\left(\frac{-\mu\tau(1+G(a,s)(1-\frac{a}{s})) - (G'(a,s)(1-\frac{a}{s}) + \frac{a}{s^2}G(a,s))}{(1+G(a,s)(1-\frac{a}{s}))^2}\right). \end{aligned}$$

Notice first that this function is continuous and the second derivative of f can be easily computed. Then, since we always have that $s > a$, the term $e^{(s\mu-\lambda)\tau}$ is always strictly positive. Therefore in order to guarantee that $|f'(s)| > 0$,

we have to check the following:

$$\left|\frac{-\mu\tau(1+G(a,s)(1-\frac{a}{s}))-(G'(a,s)(1-\frac{a}{s})+\frac{a}{s^2}G(a,s))}{(1+G(a,s)(1-\frac{a}{s}))^2}\right|>0$$

$$\left|-\mu\tau(1+G(a,s)(1-\frac{a}{s}))-(G'(a,s)(1-\frac{a}{s})+\frac{a}{s^2}G(a,s))\right|>0$$

$$\mu\tau(1+G(a,s)(1-\frac{a}{s}))+(G'(a,s)(1-\frac{a}{s})+\frac{a}{s^2}G(a,s))>0$$

which holds for all $s > a$ because summation of positive terms.

On the other hand, consider the average speed of answer stated in Definition 3.1.5: we need to make a different choice for the function $v(s)$ because of the asymptotically behavior of ASA for $s \to a^+$, as already mentioned. Indeed, it is useful to work with the inverse of the average waiting time, in order to obtain that the function $v(s)$ goes to zero when $s \to a^+$ instead of going to $+\infty$. Therefore, we want to apply the convergence methods on the following function:

$$f(s) = \frac{1}{T} - \frac{1}{\mathbb{E}(W)}$$

where the target is considered to be $1/T$, as previously explained. Notice that also for the average speed of answer we have to guarantee that hypothesis $H1$, $H2$ and $H3$ hold, in order to obtain convergence for every choice of (s_1, s_2):

- $H1$: f is continuously differentiable because $v(s)$ is also continuously differentiable for all $s > a$ since it is a probability function;
- $H2$: Equation 4.2.1 has solution for this choice of f;
- $H3$: we have to prove that the derivative f' is Lipschitz and that there exists $\rho > 0$ such that $|f'(s)| \geq \rho$ for all $s \in (s_1, s_2)$. Notice first that if $f' \in \mathcal{C}^1$ over an interval A, then f' is Lipschitz on that interval with $L \leq \sup_{s\in A} |f''(s)|$. Notice also that the second requirement of this hypothesis means that $\nexists s \in (s_1, s_2)$ such that $f'(s) = 0$, because then for continuity $\exists \rho > 0$ such that $|f'(s)| > \rho$ for all $s \in (s_1, 2)$. Therefore, we can just check that $|f'(s)| > 0$

for all $s \in (s_1, s_2)$. We can compute first the derivative f' as follows:

$$\begin{aligned} f'(s) =& - v'(s) \\ =& - \frac{d}{ds}\frac{1}{\mathbb{E}(W)} \\ =& - \frac{d}{ds}\left(s\mu\left(1-\frac{a}{s}\right)\left(1+G(a,s)\left(1-\frac{a}{s}\right)\right)\right) \\ =& \mu\left(1-\frac{a}{s}\right)\left(1+G(a,s)\left(1-\frac{a}{s}\right)\right) + \mu s\left[\frac{a}{s^2}\left(1+G(a,s)\left(1-\frac{a}{s}\right)\right)\right. \\ & \left. + \left(1-\frac{a}{s}\right)\left(G'(a,s)\left(1-\frac{a}{s}\right)+G(a,s)\frac{a}{s^2}\right)\right]. \end{aligned}$$

Notice first that this function is continuous and the second derivative of f can be easily computed. Notice also that this is a summation of all positive terms for all $s > a$, therefore the second property we need to verify is guaranteed.

Analysis of performance of convergence methods

In order to decide which is the most suitable convergence method for finding the fractional staffing level we conduct a simulation from which an analysis of performance has been made for every method. Also in this situation, the varying parameters were the forecast λ, the service rate μ, and the acceptable waiting time τ. On the other hand, for this simulation we are interested in finding the fractional number of agents, therefore the other parameter that varies in the simulation is the target T for the performance measure we are willing to achieve. For this reason, we created the simulation according to the following variation of the parameters:

- arrival rate $\lambda \in [1, 10]$;
- service rate $\mu \in [0.05, 1]$;
- acceptable waiting time $\tau \in [0.1, 1]$;
- target T, where if the target is related to the service level, then $T \in [0.5, 0.95]$; while if the target is related to the average waiting time, then $T \in [0.1, 5]$.

Notice that for each parameter, we choose 10 or 11 equidistant values in its range previously defined: 10 for λ, 11 for μ, 11 for τ, and 10 for T. We build the simulation for the service level and the average waiting time separately, in order to focus on the time execution for each type of target. Notice that the number of different combinations of parameters is different if the target is the service level or the average waiting time, because for the latter performance measure the parameter τ

is not needed. Therefore, each simulation with $T = \text{SL}$ leads to the computation of 12100 values based on all the combinations of the different parameters, while if $T = \text{ASA}$ the simulation values are 1100. Each simulation has been also made by changing the value of the interval length $l \in \{0.1, 1, 2, 4\}$. We are interested in understanding if the research of the neighborhood is computationally expensive compared to the iterations of the convergence method or not. By considering the time execution of the simulation for different values of l, we can actually verify this hypothesis, because for larger values of l, the research of the neighborhood will be faster since the target will fall sooner into it. However, having a larger neighborhood leads to a higher number of iterations for the convergence method we are applying. Therefore, it is important to analyze if the time execution increases or decreases according to the variation of l, i.e., if it is more convenient to consider faster research of the neighborhood and a higher number of iterations, or the opposite.

We discuss the time execution of the convergence methods for finding a fractional number of agents separately according to which target we want to obtain, either the service level or the average waiting time. Figure 4.3.11 presents the

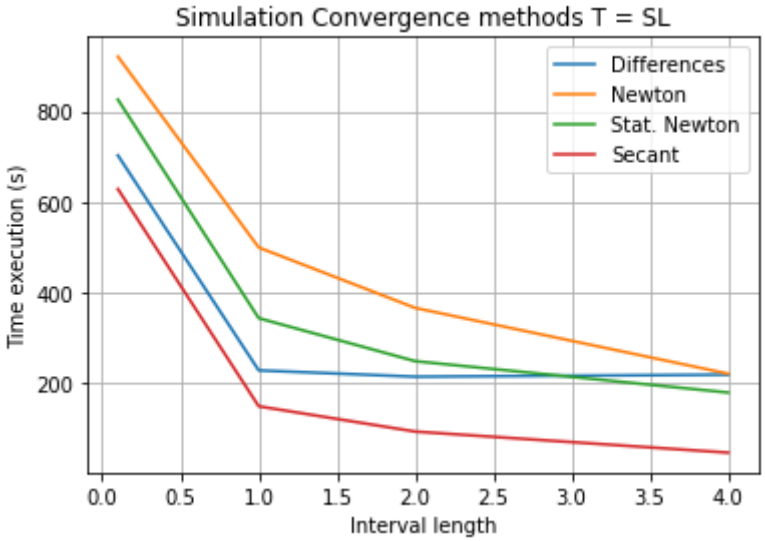

Figure 4.3.11: Time execution of convergence methods: Newton by differences, Newton, stationary Newton and secant method. The target required is the service level and the interval length l of the neighborhood on which the methods are applied varies in $\{0.1, 1, 2, 4\}$.

trend of the time execution for all the convergence methods when the interval length l of the neighborhood varies. Notice that all the different methods perform better for larger values of l. Therefore, in general, the larger the interval, the faster

will be the execution of the simulation for all methods. Focusing on larger values of l, we can see how the secant method is highly preferable since for example for $l = 4$ it takes only about 47 seconds to execute the simulation, while all the other methods take more than 170 seconds, as shown in Table 4.3.2.

Time execution (s) for $T = \text{SL}$				
Method	$l = 0.1$	$l = 1$	$l = 2$	$l = 4$
Newton by diff.	702.924	228.449	214.531	218.992
Newton	920.847	499.634	366.266	221.064
Stat. Newton	826.134	343.629	248.758	179.520
Secant	628.681	149.225	93.457	46.943

Table 4.3.2: In this table it is presented the time execution for the convergence methods for different values of the interval length l when the required target is the service level.

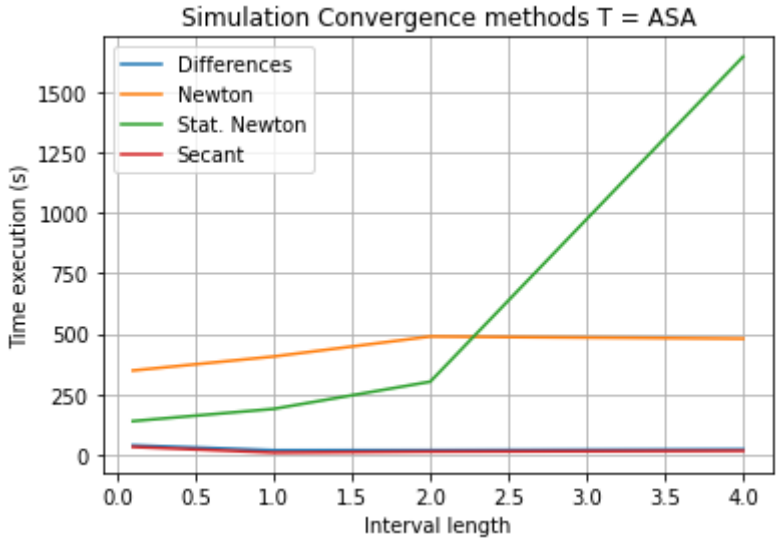

Figure 4.3.12: Time execution of convergence methods: Newton by differences, Newton, stationary Newton and secant method. The target required is the average waiting time and the interval length l of the neighborhood on which the methods are applied varies in $\{0.1, 1, 2, 4\}$.

Regarding the average waiting time, the trend of the time execution of all different methods is presented in Figure 4.3.12.

Notice that for the ASA the time execution for Newton and stationary Newton methods has an upward trend performing high values for $l = 4$ for example. Therefore, lower values of the interval length are preferable for the performance of these two methods, On the other hand, Newton by differences and secant methods perform better for all values of l. In Table 4.3.3 the time execution values are presented and it is visible how these two methods take less time to execute the simulation. In particular, the secant method for $l = 1$ takes only about 8.5 seconds to compute all the values of the simulation. Notice that the time execution is much lower in this case because the simulations are made for 1100 computations, since the parameter τ is not needed to calculate the average waiting time.

Time execution (s) for T = ASA				
Method	$l = 0.1$	$l = 1$	$l = 2$	$l = 4$
Newton by diff.	38.852	18.600	19.171	22.238
Newton	348.255	406.217	489.000	479.939
Stat. Newton	139.018	189.598	301.99	1646.620
Secant	32.182	8.559	12.709	16.028

Table 4.3.3: In this table it is presented the time execution for the convergence methods for different values of the interval length l when the required target is the average speed of answer.

It is possible to conclude that for both performance measures, service level and average waiting time, the best convergence method to find the fractional staffing level that satisfies a certain target is the secant method. This is also reasonable because by considering again Table 4.2.1 previously presented in this chapter, we can see that the secant method not only had super-linear convergence, which is quite good but also requires only one evaluation of the function f for which we want to find the zero. Notice also that for this method the computation of the derivative of the function f is not needed.

Chapter 5

Approximation via interpolation functions

The research of the fractional number of agents s that perfectly achieve certain targets for the performance measures in Erlang models can be explored with different approaches. As shown in Chapter 2, the Markov chains theory gives results for integer values of the staffing level. However, developing this theory in a continuous setting in terms of s as shown in Chapter 3 is not the only solution to this problem.

During the progress of this project, we wanted to study an alternative way to solve this problem since we realized that working in a continuous setting can be computationally expensive for the Erlang CL models, as shown in Chapter 3. Besides, for Erlang X systems, finding a continuous version of the formulas seems to be not possible with the approaches used in this project. For this reason, starting from the integer results obtained by the Markov chains formulas, we computed the fractional staffing level by interpolating these integer values. The main challenge of this part of the project consists of deciding which is the most suitable function in order to obtain a good interpolation in terms of precision and time execution.

In this chapter, after presenting the interpolation functions we decided to study, we will present how these functions have been applied to the Erlang C models and we will conduct a short analysis of performance in order to test the accuracy of the interpolation compared to the exact values computed using results in the previous chapters.

Furthermore, based on the analysis for Erlang C systems, we will discuss how to extend the interpolation approach to Erlang CL and Erlang X models.

5.1 Interpolation approaches

In order to obtain a fractional staffing level, we considered different interpolating functions that take as nodes the integer values for the staffing level, given by the Markov chain formulation.

Consider a given function f we are willing to interpolate and certain points $x_0, \ldots, x_n$ for which we know the corresponding values $f(x_0), \ldots, f(x_n)$. Then we can use different methods to construct the interpolating function p passing through the points $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$.

The first interpolation approach we tried is to build p as a linear piecewise function that for every $x \in (x_i, x_{i+1}]$

$$p(x) = f(x_{i+1}) + (x - x_{i+1}) \cdot (f(x_{i+1}) - f(x_i)) \tag{5.1.1}$$

for all $i \in \{0, \ldots, n-1\}$. This method to build the function p is called *linear interpolation.*

Another way of interpolating the function f is to use *quadratic interpolation.* In this case, the function p is again a piecewise function and in each interval it is defined as follows:

$$p(x) = ax^2 + bx + c$$

where $a, b, c \in \mathbb{R}$ are certain parameters that we need to compute. In order to do that, this interpolation approach requires either three points, or two points and the value of the derivative in one of them, for each interval.

For the first solution to find a, b and c, we need to solve the following system with three equations where we know the values of three points $(x_1, f(x_1)), (x_2, f(x_2))$ and $(x_3, f(x_3))$:

$$\begin{cases} f(x_1) = ax_1^2 + bx_1 + c \\ f(x_2) = ax_2^2 + bx_2 + c \\ f(x_3) = ax_3^2 + bx_3 + c. \end{cases} \tag{5.1.2}$$

Regarding the second solution, just two points are needed and it is necessary to know the value of the first derivative of f in one of the two points, therefore we have:

$$\begin{cases} f(x_1) = ax_1^2 + bx_1 + c \\ f(x_2) = ax_2^2 + bx_2 + c \\ f'(x_1) = 2ax_1 + b. \end{cases} \tag{5.1.3}$$

5.2 Interpolation in Erlang C systems

The interpolation functions previously described can be applied for the purpose of this project and build an alternative way of working with a continuous setting

in terms of s in Erlang models. In particular, we want to apply and test the accuracy of this approach for the Erlang C model, for which we have the exact values computed. The functions that we want to interpolate are the performance measures: service level and average speed of answer.

Consider as nodes $x_0, \ldots, x_n$ integer values of staffing level s. However, as a starting point x_0 we consider a for which we set $f(a) = 0$. With this assumption we can avoid to consider the first integer value of s before the net workload which would create positive interpolation values for a performance measure in the interval $[\lfloor a \rfloor, a]$, where we know the system is not stable.

Notice that both for the service level and the average waiting time we will consider separately the intervals $[a, \lfloor a \rfloor + 1]$ and $[\lfloor a \rfloor + 1, +\infty]$. In the interval $[a, \lfloor a \rfloor + 1]$ we always apply linear interpolation since as already mention we want to consider a as x_0, such that $f(a) = 0$. For this reason, if we apply the quadratic interpolation in this interval we will not obtain a good approximation because of the trend of the performance measures. In particular, for values of s right after a, the derivative of the service level has high values, therefore the performance measure increases steeply in the interval $[a, \lfloor a \rfloor + 1]$. On the other hand, the average waiting time has a vertical asymptote in a and it decreases sharply right after. The trend is obviously independent of the interpolation approach we want to use: to be able to make a good approximation we considered in the interval $[a, \lfloor a \rfloor + 1]$ the inverse of the average waiting time and we applied the interpolation methods on the inverse function. Notice also that by considering the inverse function we can guarantee the condition $f(a) = 0$. Even though the function we want to interpolate does not have a steep trend anymore, it is still preferable to use the linear approach because the trend now is quite flat and the quadratic approach leads to a bad approximation.

In conclusion, we always considered the linear interpolation in $[a, \lfloor a \rfloor + 1]$ because of the steep trend of the performance measures in that interval.

Consider now the region $[\lfloor a \rfloor + 1, +\infty]$. For this interval, we applied the interpolation approaches previously described in this chapter and we conducted a short analysis of performance in order to verify which provides the best approximation. Notice that in this interval we do not want to consider the inverse of the average waiting time since the ASA function is already decreasing quite smoothly without the need for any extra adjustments for our purposes.

Regarding the linear interpolation, we just considered the integer values for which we are able to compute both the service level and the average waiting time by the Markov chains formulas and we interpolated by using the function 5.1.1 previously shown.

On the other hand, in order to follow the quadratic interpolation approach we have to decide which system we want to solve between 5.1.2 and 5.1.3. Notice that

Problem 5.1.3 requires the computation of the derivative of the performance measures, which computation might be quite expensive for certain input parameters. Therefore we decided to solve Problem 5.1.2 and we obtained the following values for a, b and c:

$$\begin{cases} a = -(f(x_2) - f(x_1)) + \frac{(f(x_3)-f(x_1))}{2} \\ b = \frac{f(x_3)-f(x_1)-a(x_3^2-x_1^2)}{2} \\ c = f(x_1) - ax_1^2 - bx_1. \end{cases} \tag{5.2.1}$$

Then, for every interval $[x_1, x_2]$ such that $x_2 - x_1 = l$ we can use these parameters for the quadratic interpolation where we can consider $x_3 = x_2 + l$, in order to make an interpolation between equidistant nodes.

In our case, the nodes that we want to interpolate are the integer staffing levels computed by the Markov chains of the Erlang models. Therefore, we want to consider the parameter $l = 1$, in order to have integer equidistant values for the nodes $x_0, \ldots, x_n$.

5.3 Analysis of performance of interpolation

An analysis of performance has been conducted in order to test the accuracy of the approximation via interpolation functions. In order to make a proper comparison between the two different interpolation approaches we computed the error as the difference between the actual values from the Python implementation and the approximation via interpolation in C++.

Notice that we analyzed separately two cases: in the first scenario the number of agents is given and we want to compute the corresponding performance measure, either the service level or the average waiting time; while in the second scenario, a certain target for a performance measure is given and we want to compute the fractional number of agents needed in order to satisfy that target.

As already mentioned, in the interval $[a, \lfloor a \rfloor + 1]$ we consider for both interpolation approaches a linear interpolating function. For this reason, we decided to exclude this interval from our analysis because the error would have been the same.

Approximation of performance measures

Consider the first scenario. In this case, the simulation has been made by computing again 12100 values for different combinations of the parameters. In particular, we selected the following ranges for the parameters:

- arrival rate $\lambda \in [1, 10]$;

- service rate $\mu \in [0.05, 1]$;
- acceptable waiting time $\tau \in [0.1, 1]$;
- number of agents $s \in (a, 2a]$, where a is the net workload of the system based on the previous parameters.

The simulation has been built by choosing 10 or 11 equidistant values for each parameter in its range previously defined: 10 for λ, 11 for μ, 11 for τ, and 10 for s. As already mentioned, we excluded from the simulation all the configurations for which the staffing level s is taken in the interval $[a, \lfloor a \rfloor + 1]$, showing a certain amount of computations for each value of λ as presented in Table 5.3.1, for a total of 11451 values.

Number of computations per λ	
λ	simulation values
1	858
2	1089
3	1144
4	1177
5	1177
6	1199
7	1199
8	1199
9	1199
10	1210

Table 5.3.1: This table shows how many values has been computed in the simulation for each value of λ after removing all the values for which $s \in [a, \lfloor a \rfloor + 1]$.

The first section of Appendix B presents the graphs showing the errors between the actual values computed by the Python implementation and the interpolation approximation when a certain number of agents s is given and we want to compute the corresponding service level. The errors are presented separately according to the selected value of λ in the simulation. It is visible in all 10 different graphs that the error of the linear interpolation is higher than the exact computation compared to the error of the quadratic interpolation. Since the service level is a concave function, we expected to have an underestimation of the performance measure for a given number of agents with the linear interpolation approach. Besides, we can notice that the higher error reached is by the linear interpolation when the arrival rate λ is 1, showing a difference of over 10%. On the other hand, the higher errors reached by the quadratic interpolation are about 4% for λ equal to 1, 2

and 3. Overall, in order to obtain a service level for a certain given number of agents, the quadratic approximation always gives lower errors compared to the linear interpolation.

On the other hand, in the second section of Appendix B, it is possible to visualize the same error graphs in the scenario where we want to compute the average waiting time when a certain staffing level is given. Notice that in this case the average waiting time is a convex function, therefore the linear approximation is always overestimating the performance measure for a certain value of s. Indeed, it is possible to see how the error for the linear interpolation becomes negative and assumes very low values: for example for $\lambda = 1$, we obtain an error of about -140. For all the other values of λ, the error does not reach such low values, however, it is still evident the difference with the error obtained by the quadratic interpolation, which on average is preferable to the one computed by the linear approach.

Approximation of fractional staffing level

In the next sections of Appendix B, it is possible to find other graphs that represent the approximation of the interpolation functions to find fractional staffing levels whenever a target for a performance measure is given. The simulation of this part consisted of computing again several values according to all the different combinations of the following parameters:

- arrival rate $\lambda \in [1, 10]$;
- service rate $\mu \in [0.05, 1]$;
- acceptable waiting time $\tau \in [0.1, 1]$;
- target T, where if the target is related to the service level, then $T \in [0.5, 0.95]$; while if the target is related to the average waiting time, then $T \in [0.1, 5]$.

We consider again for this simulation 10 values for λ, 11 for μ, 11 for τ, and 10 for the target T. When the required target is the service level, the simulation computes 12100 values, while if the target is the average waiting time then the number of computations is 1100 since the parameter τ is not required.

Also for this analysis of performance, we collected the results per different values of λ and we considered only the result for all staffing levels greater than $\lfloor a \rfloor + 1$ in order to make a proper comparison between the two interpolation approaches.

First, the approximation for each λ is presented in the case when the target required is the service level and we want to compute the corresponding number of agents to exactly achieve that target. Notice that with both interpolation approaches, the approximation overestimates the number of agents since the error

between the exact values and the interpolation is negative. However, it is shown that for each λ, the maximum error obtained in the simulation for the linear interpolation is always almost double the error of the quadratic interpolation, which overall approximates better the staffing values.

Secondly, the results of these approximations when the average speed of answer is the required target are also included in Appendix B. Regarding this case, we considered the absolute error between the exact values and the interpolation approximation due to the fact that the quadratic interpolation is overstaffing, while the linear functions underestimate the exact solution. For this reason, by computing the absolute error it is possible to actually make a comparison between the two approaches. Also in this case, the quadratic interpolation computes staffing results that are closer to the exact values compared to the results given from the linear approximation. In particular, notice that for values of λ greater than 5 the absolute error given by the linear interpolation is on average up to three times the quadratic one.

5.4 Results of the interpolation approximations

As previously analyzed in this chapter, it is possible to deduce that the quadratic approach for the interpolation of the performance measures is always better than the linear approach. The quadratic interpolation gives lower values for the error for both the service level and the average waiting time. Besides, it performs better also for all the different values of λ considered for our simulations.

Also, the quadratic interpolation approach shows better approximation for both scenarios: when the number of agents is given and we want to compute the performance measure, and when from a certain target we want to compute the corresponding staffing level required to satisfy that specific target.

As already mentioned, all the results of the simulation are presented in Appendix B, divided into sections according to the considered scenario.

Table 5.4.1 shows the time execution of every simulation made to test the interpolation approaches when the number of agents is given and we want to compute the corresponding values for the performance measures. Notice that in this simulation for every combination of parameters we computed both the service level and the average waiting time.

In Table 5.4.2, we show the time execution for both the linear and the quadratic interpolation when the given target is the service level and we want to obtain the corresponding fractional staffing level.

Finally, in Table 5.4.3 we show the time execution for both the linear and quadratic interpolation when the given target is the average waiting time and the goal is to compute the fractional staffing level corresponding to that target. Notice

Time execution (s)	
Approach	Time
Linear	1.471s
Quadratic	1.603s

Table 5.4.1: For given the numbers of agents, both the service level and the average waiting time have been computed with a simulation of 12100 values.

Time execution (SL)	
Approach	Time
Linear	0.731s
Quadratic	0.981s

Table 5.4.2: For given targets (SL), we compute the corresponding fractional number of agents in a simulation of 12100 values.

that for this simulation we considered 1100 different combinations of parameters instead of 12100 because the parameter τ is not required whenever the given target is the average waiting time.

Time execution (ASA)	
Approach	Time
Linear	0.061s
Quadratic	0.057s

Table 5.4.3: For given targets (ASA), we compute the corresponding fractional number of agents in a simulation of 1100 values.

Chapter 6

Final results and conclusion

In this final chapter, the key part of the conclusive results consists of how we can find useful insights from the approaches used for Erlang C in order to extend this research to Erlang CL and Erlang X systems. The conclusion of this research will be also presented by discussing limitations and any further studies that can be made on this topic.

6.1 Discussion and conclusion

The goal of this research is to identify a suitable method to compute fractional staffing levels related to certain targets for the performance measures in Erlang models. This project mainly determines good solutions for the Erlang C models, while those for Erlang CL and Erlang X are not entirely discussed. The solution we want to propose for these two models is to make use of the knowledge obtained from the analysis of Erlang C and extend it to the other models. In particular, as already mentioned we do not have explicit continuous formulas for Erlang X, while Erlang CL presents computations that are quite challenging to be implemented in C++. Therefore, we want to use the best method found for Erlang C and apply it to the other two models.

Recall from Chapter 4 that the best method to compute the performance measures for Erlang C is by using the Simpson rule as a numerical integration method with a setting of 10000 steps, where the time of execution of 12100 computations of our simulation is approximately around 36 seconds. Later in the same chapter, we concluded that the best convergence method to find the fractional staffing level when a certain target is given is the secant method. In terms of accuracy, the results are optimal because the theorems of convergence guarantee finding the optimal solution. On the other hand, in terms of time execution the secant method applied in a neighborhood of length $l = 1$ takes approximately 8 seconds to reach

the solution for 1100 different values of the simulation.

Subsequently in Chapter 5, we showed that via quadratic interpolation it is possible to obtain the values of the performance measures or the fractional staffing level when a certain target is given within a small fraction of seconds as shown at the end of Chapter 5. These values of time execution guarantee a quick performance of this approach to find fractional staffing levels. In terms of accuracy, we presented in Appendix B the errors between the exact values and the interpolation approximation. The exact values are obtained by the study and the implementation of the continuous re-formulation of the Erlang C formulas. Notice that overall the quadratic interpolation performs better compared to the linear approach as already mentioned. Therefore, the quadratic interpolation guarantees a fast and good approximation of the fractional staffing level for Erlang C models.

In conclusion, by using the results obtained in Chapter 3 and Chapter 4 we are able to find a faster approach for the research of the fractional number of agents that exactly meets certain required targets for Erlang C. As already mentioned, we can finally then extend this approach to other more complex models, such as Erlang CL and Erlang X, which represent contact centers in a more realistic manner.

This conclusion allows us to handle staffing decisions with fractional values avoiding one rounding in the capacity management and eventually decreasing the staffing costs in the WFM process.

6.2 Limitations and future research

In this project, there have been many challenges to face in order to find optimal methods both from a theoretical and practical point of view.

One of the biggest limitations of this project is the lack of literature on this topic. Despite the extensive available literature on Erlang models, there appears to be an omission in articles that concerns the potential application of these models in a continuous setting. This inevitably leads to a scarcity of sources supporting this project, particularly from a theoretical perspective. However, the lack of studies in this specific area of Erlang models gives space for many innovative insights for future research.

In this research, consistent theoretical results have been discovered for Erlang C models, by proving the equality between the discrete and the continuous formulas. Once the continuous formulas were proven to be consistent, we faced the implementation part of the project which required knowledge of numerical analysis. The analysis and the methods used in the project can be extended and improved to more advanced techniques to solve problems of numerical integration and convergence methods. The results obtained from the implementation of the

continuous formulas are consistent with the main goal of the project since they allow us to compute the performance measures and test the accuracy of the interpolation methods. However, the difference in terms of time execution for these methods and the interpolation approaches is quite consistent. For this reason, the margin of improvement especially in terms of speed of convergence is quite large. Regarding the accuracy of the results, the numerical integration in C++ can be improved by using different methods that result to have lower errors compared to the Python computations. On the other hand, the convergence methods already reach the optimal results owing to the theorems that guarantee the convergence of each iterative method.

In Chapter 3, we also present consistent formulas for Erlang CL systems. Due to their complex expression, it was not possible to find suitable methods to implement these formulas in C++. Therefore, a wide area for further research consists of finding new approaches for the implementation of the Erlang CL formulas. In particular, the most difficult performance to implement is the service level, since it requires the computation of a double integral as presented in Proposition 3.2.2. In this research, we aimed to compute this double integral by using the numerical integration methods presented in Chapter 4 in order to calculate in each variable of integration the corresponding area. However, this approach resulted to be not precise enough and quite slow. In order to reach accuracy, the partition for each variable of integration needed to be thinner than for Erlang C models. Subsequently, this led to a large number of computations because not only 1000, 10000, or 100000 were not enough as a number of steps, but also the computation has to be done in two dimensions, which increased the time execution of the methods quadratically. For this reason, we concluded that the numerical integration methods presented for Erlang C are not suitable for the implementation of the Erlang CL performance measures.

Another large area of improvement for this research is related to the Erlang X systems. Due to the complexity of the models, it was not possible to find appropriate formulas for a continuous system in terms of staffing level. The approach used during this research that aimed to discover the continuous formulas for Erlang X was to consider first a more simple model: Erlang A. The main difference between Erlang X and Erlang A is that the latter is an infinite capacity system, while in Erlang X models we consider the blocking probability in case the system is already complete. The idea behind our approach was to consider another equality between a summation and an integral, as in Theorem 3.1.1 for Erlang C, for the Erlang A models. Once the formulas for Erlang A were computed, the next step was to rewrite the finite Erlang X system in terms of the infinite formulas. This approach required more studies and proofs than the research for Erlang C systems. For this reason, we could not conclude any specific reformulation for Erlang X, however, it

represents a wide area for further research on this topic.

Appendix A

Particular case of Erlang CL formulas

In Chapter 2 and Chapter 3, we describe respectively the discrete and continuous formulas of Erlang CL models when $a \neq s$. Differently from Erlang C, in Erlang CL systems all values of s can be used to compute the stationary distribution and the performance measures. Since the capacity of the queue is finite, the system is always stable and there are no conditions that need to be guaranteed. For this reason, we computed both the stationary distribution and the performance measures for all values of $s \in S$. The only particular case that was not previously discussed is when $a = s$. We decided to analyze this case separately because for this value of s the stationary distribution assumes different expressions.

Moreover, also the continuous formulation of the Erlang CL models changes whenever the number of agents s is equal to the net workload.

We analyze in this appendix this particular case and how the different formulas change.

A.1 Stationary distribution

First, we derive the stationary distribution of Erlang CL systems when $a = s$ for both discrete and continuous expressions.

Discrete formulas

Recall that in Chapter 2 we define the stationary distribution of the Markov chain for Erlang CL systems as follows:

- if $x < s$:

$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right]^{-1};$$

- if $x \geq s$:

$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}\frac{1-\left(\frac{a}{s}\right)^{N+1-s}}{1-\frac{a}{s}}\right]^{-1}.$$

for all $x \in S = \{0, 1, \ldots, N\}$. This expression is obtained by using the equality $\sum_{j=m}^{n} x^j = \frac{x^{n+1}-x^m}{x-1}$ with $x \neq 1$, where $x = \frac{a}{s}$. As previously mentioned, the case we want to discuss in this appendix is when $a = s$, i.e., when $x = 1$. Therefore, this well-known summation is not applicable. Consider then the stationary distribution obtained from the Balance Equation and the normalization as for the general case in Chapter 2. Hence, we have

$$\begin{aligned}
\sum_{x=0}^{N}\pi(x) &= \sum_{x=0}^{s-1}\pi(x) + \sum_{x=s}^{N}\pi(x) \\
&= \sum_{x=0}^{s-1}\frac{a^x}{x!}\pi(0) + \sum_{x=s}^{N}\left(\frac{a}{s}\right)^{x-s}\frac{a^s}{s!}\pi(0) \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\left(\frac{a}{s}\right)^{-s}\sum_{x=s}^{N}\left(\frac{a}{s}\right)^{x}\right].
\end{aligned}$$

Notice that in this case, since $a = s$ we have the following:

$$\begin{aligned}
\sum_{x=0}^{N}\pi(x) &= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}\left(\frac{a}{s}\right)^{-s}\sum_{x=s}^{N}\left(\frac{a}{s}\right)^{x}\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}1^{-s}\sum_{x=s}^{N}1^x\right] \\
&= \pi(0)\left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}(N-s)\right] \\
\implies \pi(0) &= \left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}(N-s)\right]^{-1}.
\end{aligned}$$

Therefore, the stationary distribution when $a = s$ is the following:

- if $x < s$:

$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}(N-s)\right]^{-1};$$

- if $x \geq s$:

$$\pi(x) = \left(\frac{a}{s}\right)^{x-s} \frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}(N-s)\right]^{-1}$$
$$= \frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}(N-s)\right]^{-1}$$

for all $x \in S = \{0, 1, \ldots, N\}$. Therefore we have the following proposition:

Proposition A.1.1. *Given a state space $S = \{0, 1, 2, \ldots, N\}$ and $s = a$, the Erlang CL model defined on this state space has stationary distribution π defined as follows:*

- *if $x < s$:*

$$\pi(x) = \frac{a^x}{x!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}(N-s)\right]^{-1};$$

- *if $x \geq s$:*

$$\pi(x) = \frac{a^s}{s!}\left[\sum_{j=0}^{s-1}\frac{a^j}{j!} + \frac{a^s}{s!}(N-s)\right]^{-1};$$

for all $x \in S$.

Continuous formulas

After computing the discrete expression of the stationary distribution stated in Proposition A.1.1, we can analyze the continuous formulas. Notice that the function $G(a, s)$ defined in Theorem 3.1.1 does not change its behavior when $s = a$, therefore we do not need to consider it differently. By using the function $G(a, s)$, we can write the probability $\pi(0)$ as follows:

$$\pi(0) = \left[\sum_{x=0}^{s-1}\frac{a^x}{x!} + \frac{a^s}{s!}(N-s)\right]^{-1}$$
$$= \frac{a^s}{s!}\left[\sum_{x=0}^{s-1}\frac{s!a^{x-s}}{x!} + N - s\right]^{-1}$$
$$= \frac{a^s}{s!}\left[G(a,s) + N - s\right]^{-1}.$$

Therefore we obtain the following proposition:

Proposition A.1.2. *Given a state space $S = \{0, 1, 2, \ldots, N\}$ and $s = a$, the Erlang CL model defined on this state space has stationary distribution π defined as follows:*

- *if $x < s$:*
$$\pi(x) = \frac{a^x}{x!}\left[\frac{a^s}{s!}\left(G(a,s) + N - s\right)\right]^{-1};$$
- *if $x \geq s$:*
$$\pi(x) = \left(\frac{a}{s}\right)^{x-s}\left[G(a,s) + N - s\right]^{-1};$$

for all $x \in S = \{0, 1, \ldots, N\}$, where $G(a, s)$ is defined as in Theorem 3.1.1.

A.2 Service level

The first performance measure of Erlang CL we want to compute for the case $a = s$ is the service level. Notice that the service level stated in Proposition 2.2.2, i.e., the discrete formula, does not change its expression for this case because the only terms that need to be modified are the probabilities $\pi(x)$ of the formula. Therefore, the expression for this particular case is quite straightforward by using Proposition A.1.1.

Regarding the continuous expression for the service level stated in Proposition 3.2.2, we need to consider a different expression as stated in the following proposition:

Proposition A.2.1. *In Erlang CL systems, if $s = a$ the service level can be written as*
$$\mathrm{SL} = \frac{1}{1 - \pi(N)}\left[1 - \left(B(s) + \pi(N)\right)\right]$$
with
$$\begin{aligned} B(s) =& \frac{(\mu s\tau)^{N-s+1}}{(N-s-1)!}\frac{a^N}{s!}\pi(0)e^{-\mu s\tau}\cdot \\ & \cdot \int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}\int_0^{+\infty} e^{-\mu s\tau(1+t)u}(1+u)^{N-s-1}dudt \end{aligned}$$
where the function $G(a, s)$ is defined as in Theorem 3.1.1.

Proof. Recall first the expression for the service level in Erlang CL systems found in the previous chapter stated in Proposition 2.2.2:
$$\mathrm{SL} = \frac{1}{1 - \pi(N)}\left[1 - \left(\sum_{i=0}^{N-s-1}\pi(s+i)\sum_{k=0}^{i}\frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} + \pi(N)\right)\right].$$

Observe that we showed by Proposition A.1.2 the continuous formulas for the stationary distribution when $s = a$, therefore we can already consider the term $\pi(N)$ in the previous expression as a continuous function of the number of agents. Then, we are interested in proving the following equality

$$B(s) = \sum_{i=0}^{N-s-1} \pi(s+i) \sum_{k=0}^{i} \frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} \qquad (\star\star)$$

where the function $B(s)$ is defined in the statement of the proposition. We can start with the right-hand side of Equation $(\star\star)$ by using the definition of the function $G(a, s)$ given in Theorem 3.1.1 by taking as input values $\mu s\tau$ and $i + 1$ respectively. Then,

$$\begin{aligned}
\text{RHS} &= \sum_{i=0}^{N-s-1} \pi(s+i) \sum_{k=0}^{i} \frac{(\mu s\tau)^k e^{-\mu s\tau}}{k!} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \sum_{k=0}^{i} \frac{(\mu s\tau)^k}{k!} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \frac{(\mu s\tau)^{i+1}}{(i+1)!} \sum_{k=0}^{i} \frac{(\mu s\tau)^{k-(i+1)}(i+1)!}{k!} \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \frac{(\mu s\tau)^{i+1}}{(i+1)!} G(\mu s\tau, i+1) \\
&= \sum_{i=0}^{N-s-1} \pi(s+i) e^{-\mu s\tau} \frac{(\mu s\tau)^{i+1}}{(i+1)!} \int_0^{+\infty} (i+1) e^{-\mu s\tau t} (1+t)^i dt.
\end{aligned}$$

Now since the summation is finite we can exchange it with the integral and obtain the following:

$$\begin{aligned}
\text{RHS} &= e^{-\mu s\tau} \int_0^{+\infty} e^{-\mu s\tau t} \sum_{i=0}^{N-s-1} \pi(s+i) \frac{(\mu s\tau)^{i+1}}{(i+1)!} (i+1)(1+t)^i dt \\
&= \mu s\tau e^{-\mu s\tau} \int_0^{+\infty} e^{-\mu s\tau t} \sum_{i=0}^{N-s-1} \pi(s+i) \frac{(\mu s\tau(1+t))^i}{i!} dt.
\end{aligned}$$

Notice that Proposition A.1.2 shows that for $x \geq s$ the stationary distribution is constant and it is defined as $\pi(x) = \frac{a^s}{s!}\pi(0)$, where we already know that $\pi(0)$ is a

continuous function of the staffing level s. Therefore,

$$\begin{aligned}\text{RHS} &= \mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\sum_{i=0}^{N-s-1}\frac{a^s}{s!}\pi(0)\frac{(\mu s\tau(1+t))^i}{i!}dt\\ &= \frac{a^s}{s!}\pi(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\sum_{i=0}^{N-s-1}\frac{(\mu s\tau(1+t))^i}{i!}dt.\end{aligned}$$

By considering again the definition of the function $G(a,s)$ of Theorem 3.1.1 with input values $\mu s\tau(1+t)$ and $N-s$ respectively, we can rewrite

$$\begin{aligned}\text{RHS} =&\frac{a^s}{s!}\pi(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}\frac{(\mu s\tau(1+t))^{N-s}}{(N-s)!}G(\mu s\tau(1+t),N-s)dt\\ =&\frac{(\mu s\tau)^{N-s}}{(N-s)!}\frac{a^s}{s!}\pi(0)\mu s\tau e^{-\mu s\tau}\int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}G(\mu s\tau(1+t),N-s)dt\\ =&\frac{(\mu s\tau)^{N-s+1}}{(N-s)!}\frac{a^N}{s!}\pi(0)e^{-\mu s\tau}\cdot\\ &\cdot\int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}\int_0^{+\infty}(N-s)e^{-\mu s\tau(1+t)u}(1+u)^{N-s-1}dudt\\ =&\frac{(\mu s\tau)^{N-s+1}}{(N-s-1)!}\frac{a^N}{s!}\pi(0)e^{-\mu s\tau}\cdot\\ &\cdot\int_0^{+\infty} e^{-\mu s\tau t}(1+t)^{N-s}\int_0^{+\infty} e^{-\mu s\tau(1+t)u}(1+u)^{N-s-1}dudt\end{aligned}$$

which proves the equality of the Equation $(\star\star)$. □

A.3 Average waiting time

The second performance measure of Erlang CL models that changes whenever $a = s$ is the average speed of answer. Notice that the discrete formula for the ASA is given by Proposition 2.2.3, which does not need to be modified except for the terms of the form $\pi(x)$. Therefore, the discrete average waiting time when $a = s$ is straightforward by using Proposition A.1.1.

On the other hand, for the continuous formula of the ASA we need to compute the performance measure differently as stated in the following proposition:

if $a = s$:

Proposition A.3.1. *In Erlang CL systems, if $s = a$ the average speed of answer can be written as follows:*

$$\text{ASA} = \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s}\cdot\frac{(N-s)(N-s+1)}{2};$$

where $\pi(0)$ and $\pi(N)$ can be written as continuous functions of s as in Proposition A.1.2.

Proof. Recall first the expression of ASA given in Proposition 2.2.3

$$\text{ASA} = \frac{1}{1-\pi(N)} \sum_{i=0}^{N-s-1} \frac{i+1}{s\mu} \pi(s+i).$$

Consider the stationary distribution previously found $\pi(s+i) = \frac{a^s}{s!}\pi(0)$ in Proposition A.1.2, where $\pi(0)$ is a continuous function of s. Then,

$$\begin{aligned}
\text{ASA} &= \frac{1}{1-\pi(N)} \sum_{i=0}^{N-s-1} \frac{i+1}{s\mu} \frac{a^s}{s!} \pi(0) \\
&= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s} \sum_{i=0}^{N-s-1} (i+1) \\
&= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s} \sum_{j=1}^{N-s} j \\
&= \frac{\frac{a^s}{s!}\pi(0)}{(1-\pi(N))\mu s} \cdot \frac{(N-s)(N-s+1)}{2}
\end{aligned}$$

since we have the known result $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$. □

A.4 Occupancy

The last performance measure is the occupancy defined as follows:

$$\text{OCC} = \frac{a}{s}(1-\pi(N)).$$

When $a = s$, we just need to change the term $\pi(N)$ which can be obtained straightforwardly from Proposition A.1.1 and Proposition A.1.2 for the discrete and the continuous case respectively.

Appendix B

Interpolation graphs

In this appendix, we will present some graphs showing the error between the actual values computed with the Python implementation of the model and the interpolation approximation implemented in C++. For every graph, we can see in blue the error of the linear approximation and in orange the error of the quadratic approximation.

We show first the service level when the number of agents s is given; subsequently, the average waiting time for a given number of agents; and finally the fractional staffing when first the service level is given and then when the given performance measure is the average waiting time.

B.1 Service level for a given number of agents

In this section, we will show the error of the service level between the actual values and the interpolation among integer values of s.

Each graph shows the error for a specific value of the parameter λ considered in the simulation, i.e., $\lambda \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

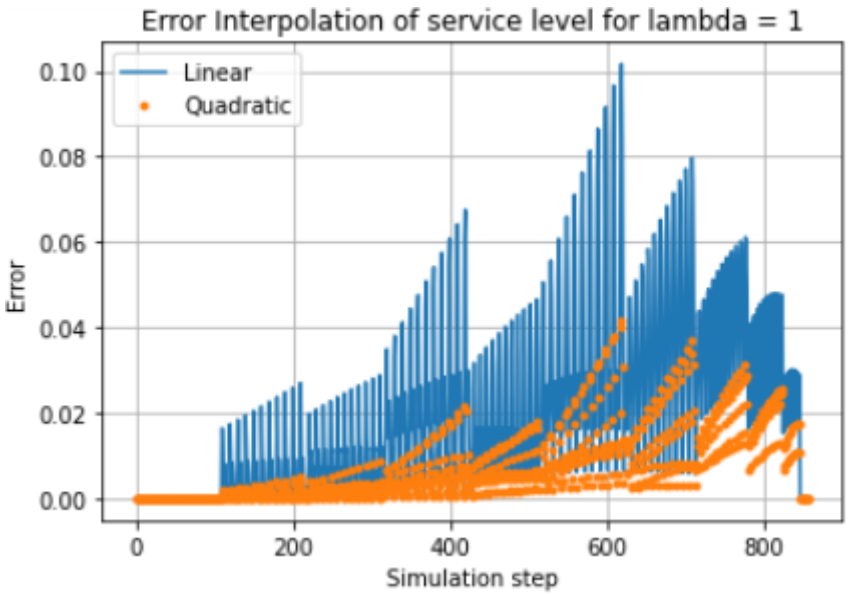

Figure B.1.1: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 1$.

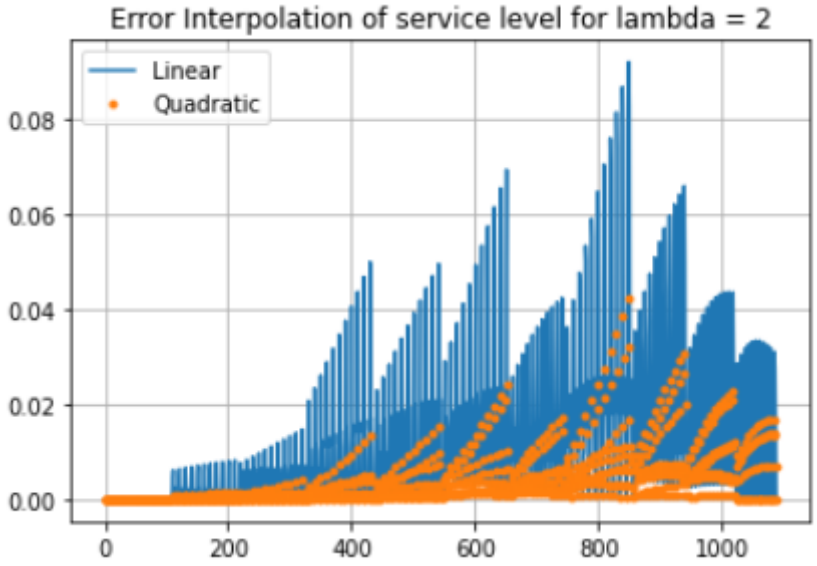

Figure B.1.2: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 2$.

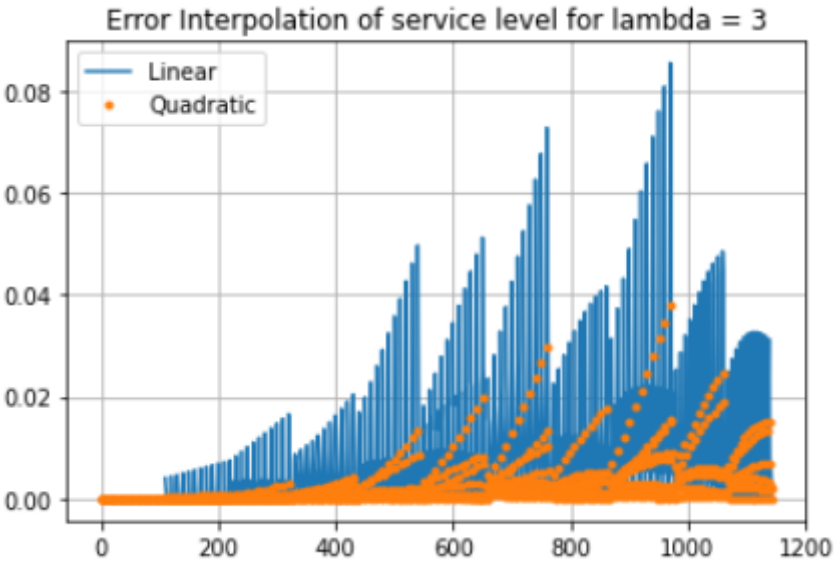

Figure B.1.3: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 3$.

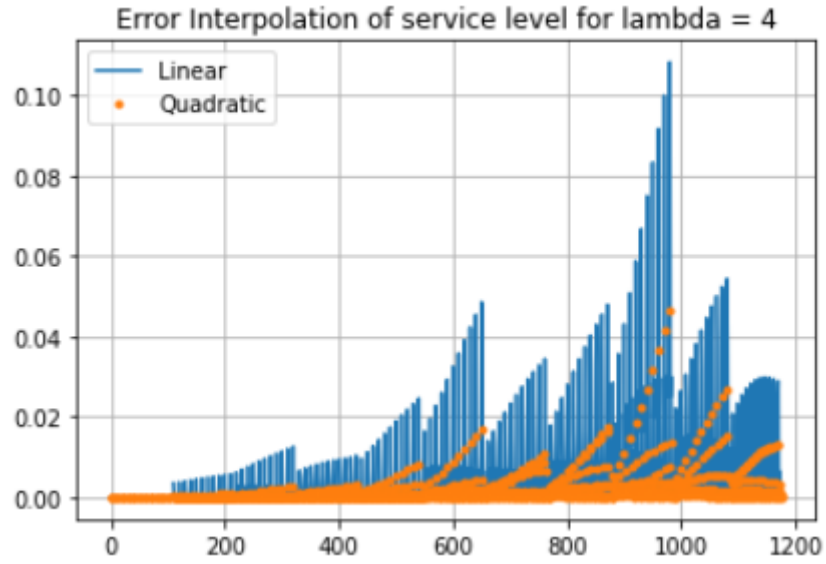

Figure B.1.4: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 4$.

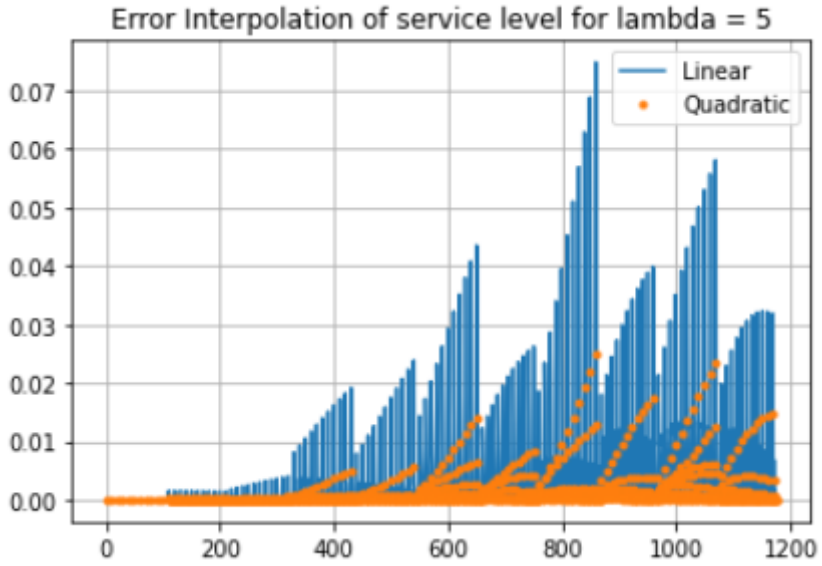

Figure B.1.5: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 5$.

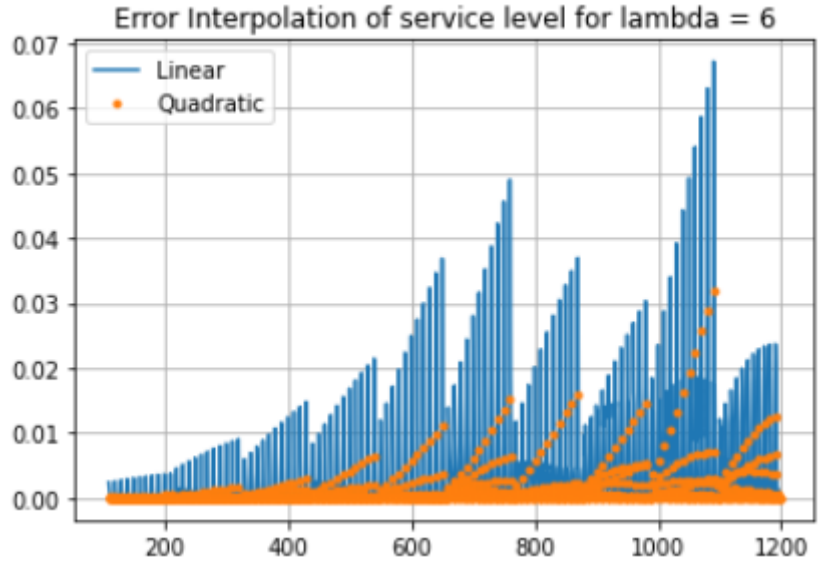

Figure B.1.6: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 6$.

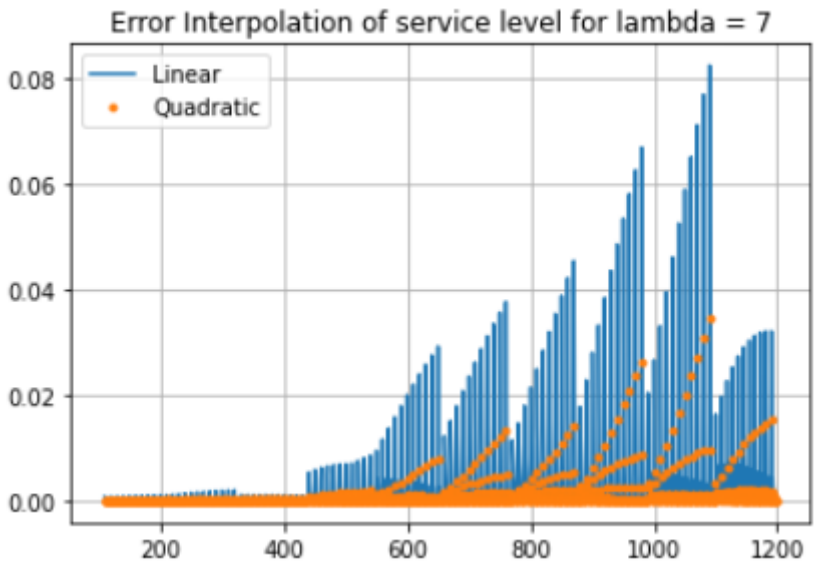

Figure B.1.7: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 7$.

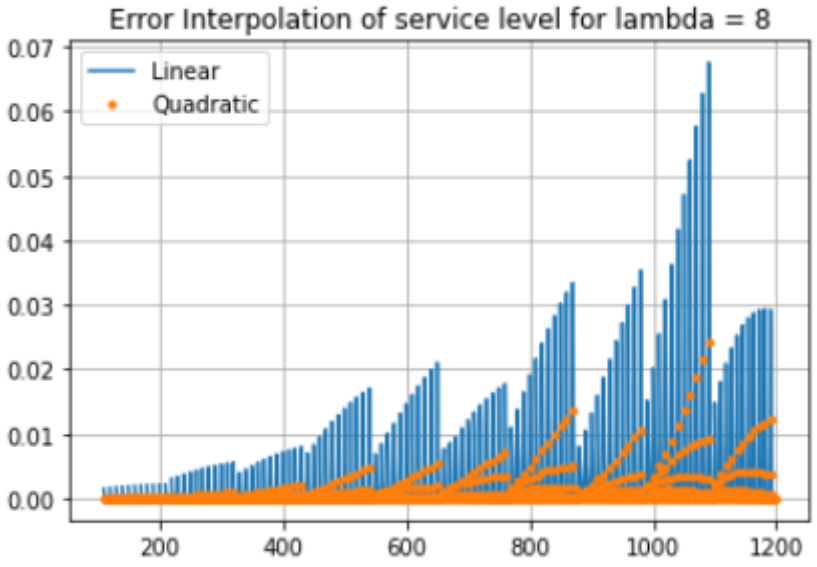

Figure B.1.8: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 8$.

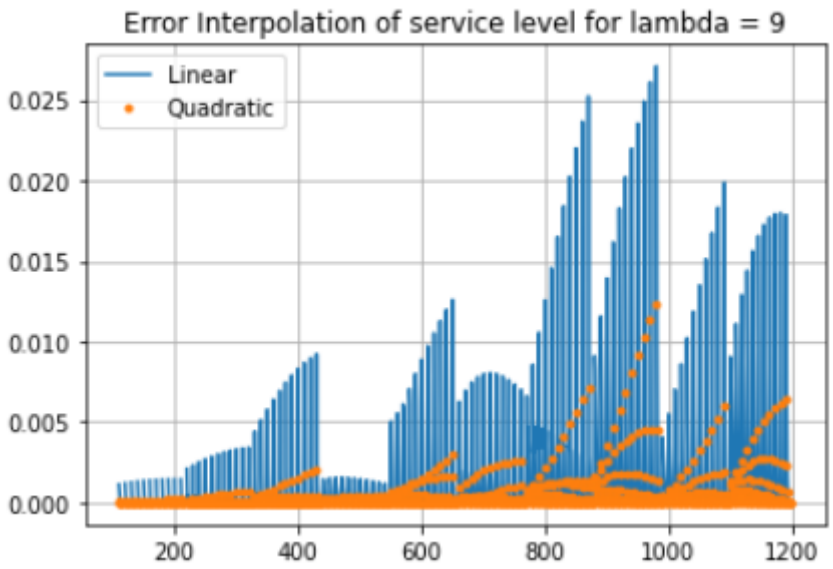

Figure B.1.9: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 9$.

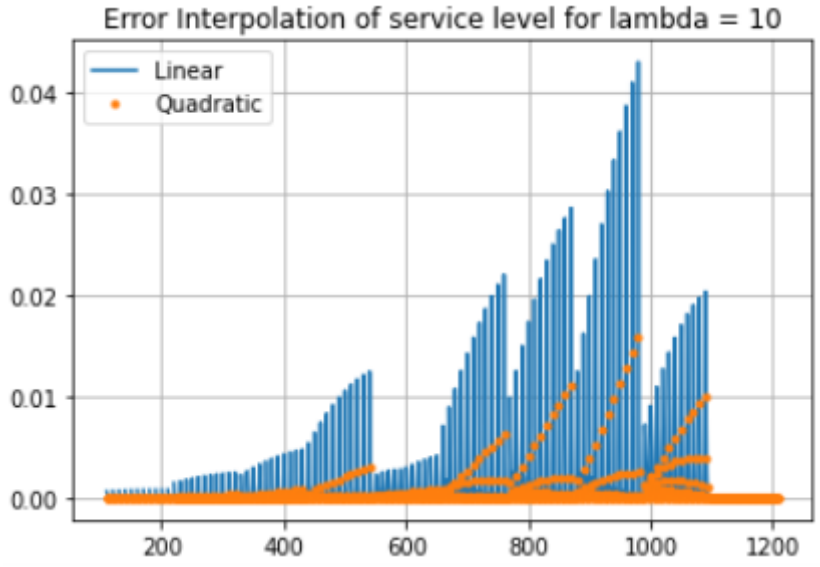

Figure B.1.10: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 10$.

B.2 Average waiting time for a given number of agents

In this section we will show the error of the average waiting time between the actual values and the interpolation among integer values of s.

Each graph shows the error for a specific value of the parameter λ considered in the simulation, i.e., $\lambda \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

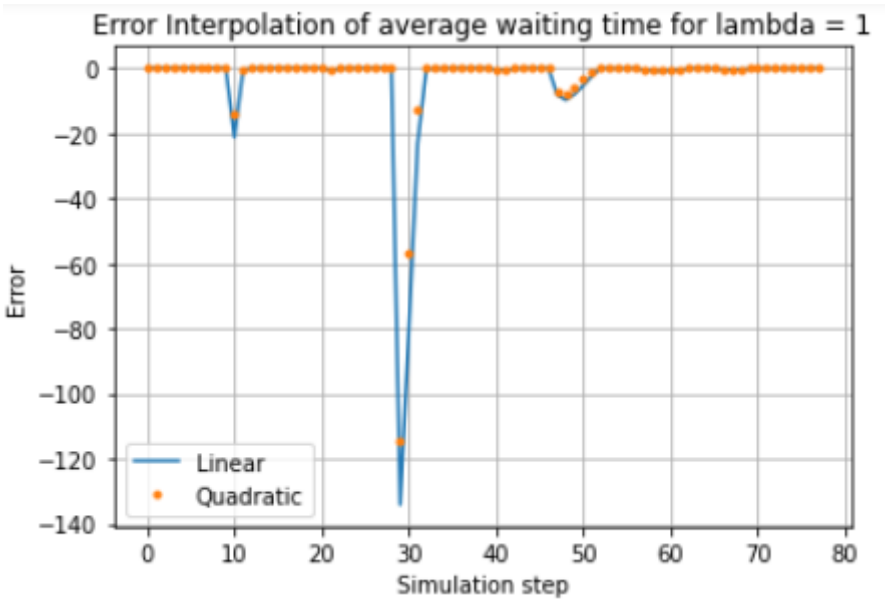

Figure B.2.1: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 1$.

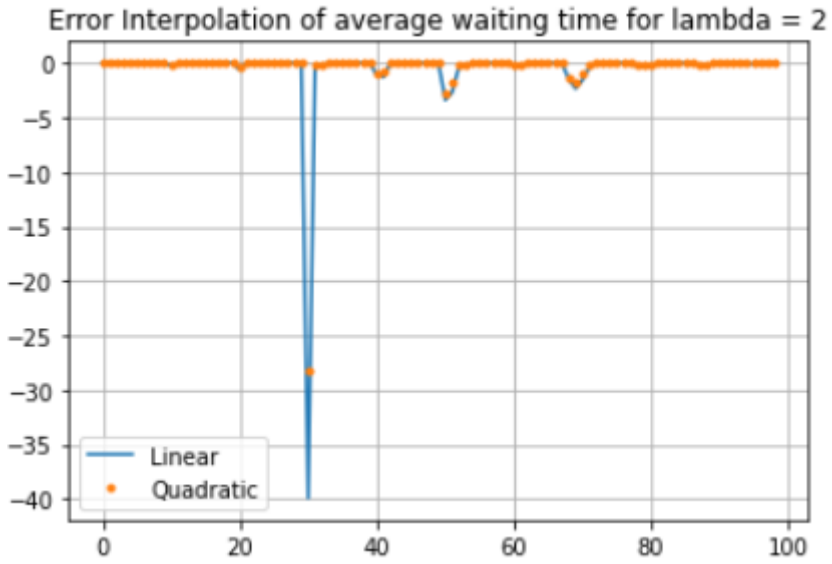

Figure B.2.2: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 2$.

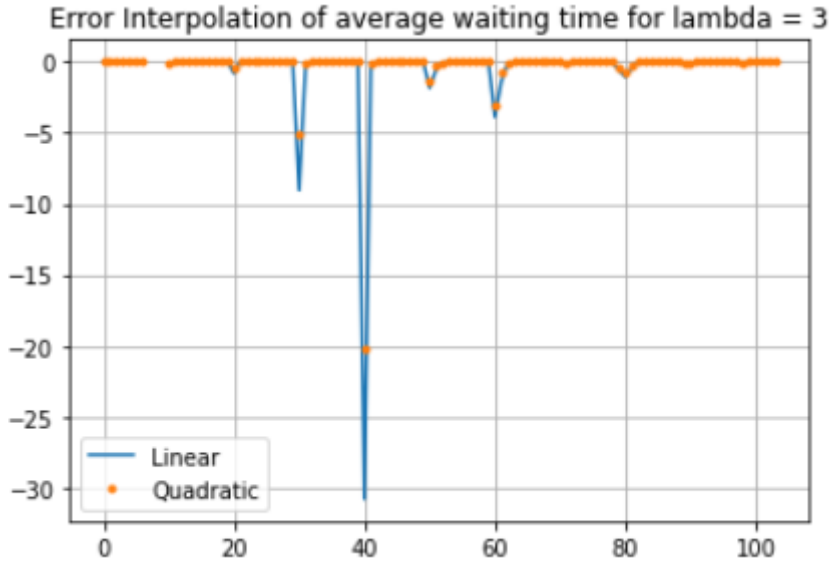

Figure B.2.3: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 3$.

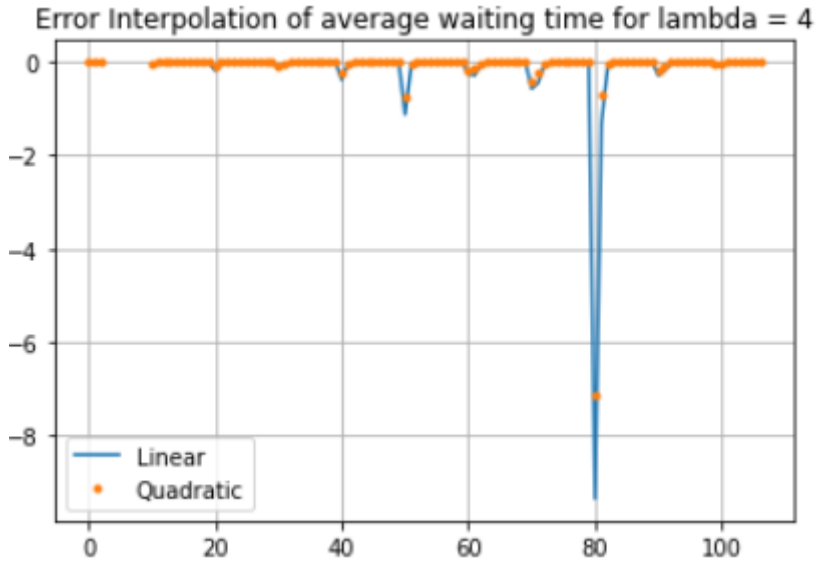

Figure B.2.4: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 4$.

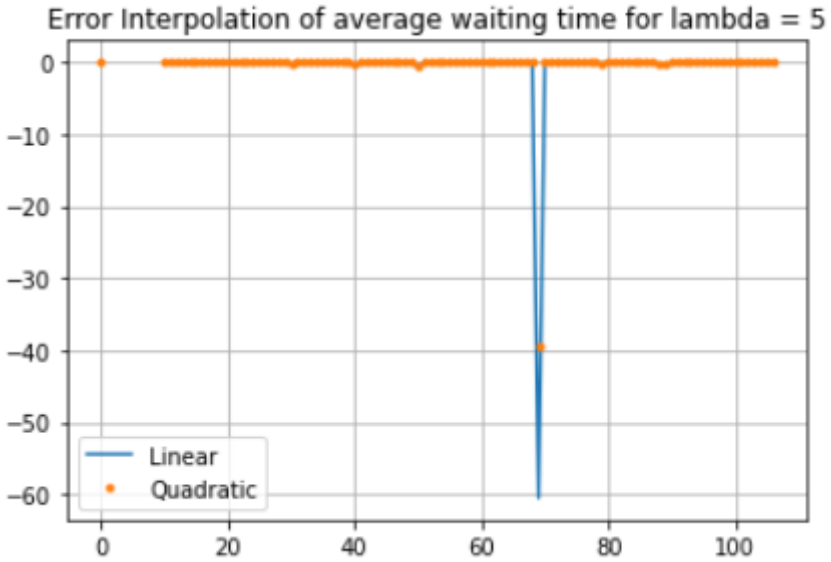

Figure B.2.5: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 5$.

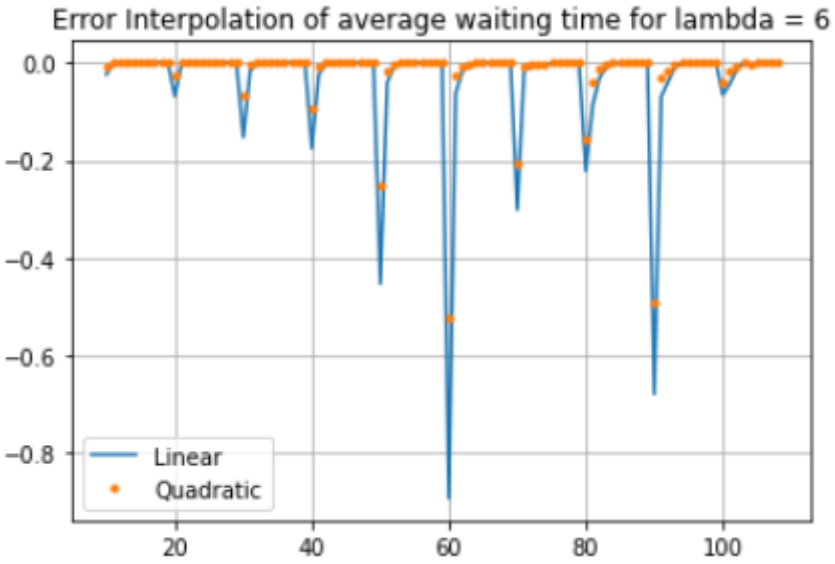

Figure B.2.6: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 6$.

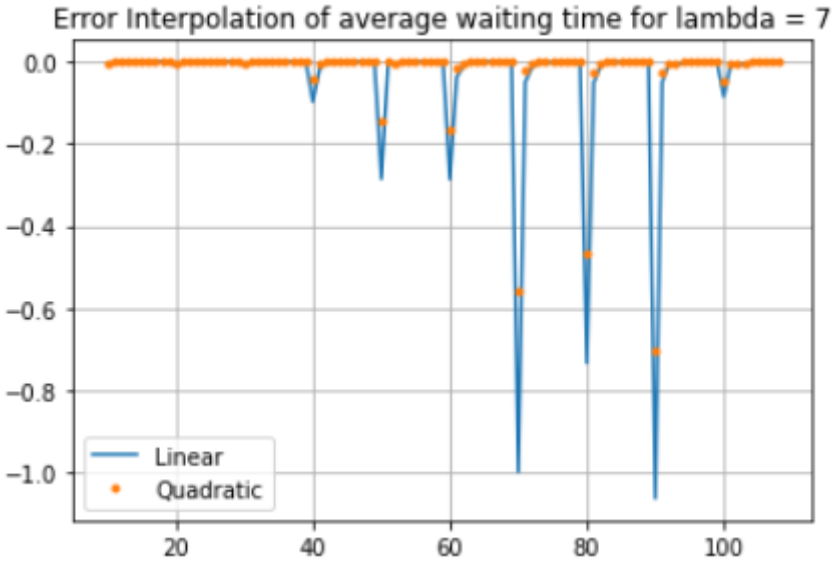

Figure B.2.7: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 7$.

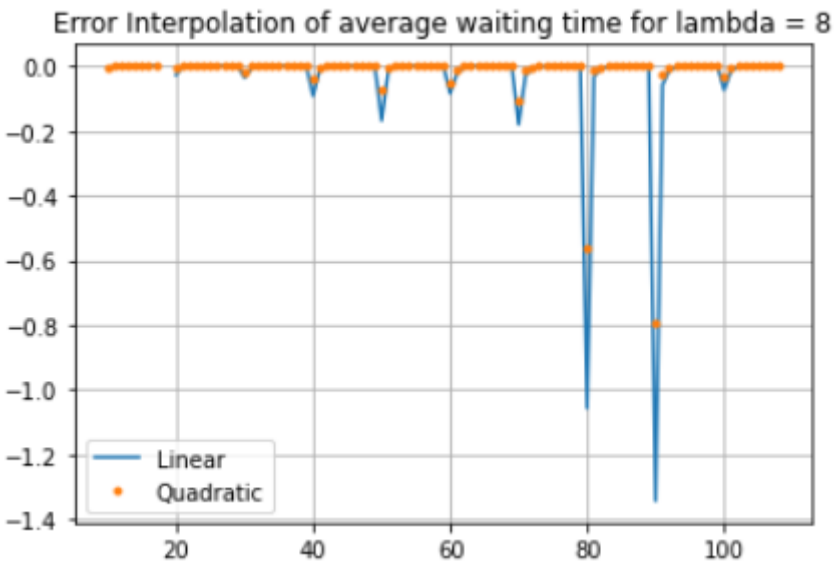

Figure B.2.8: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 8$.

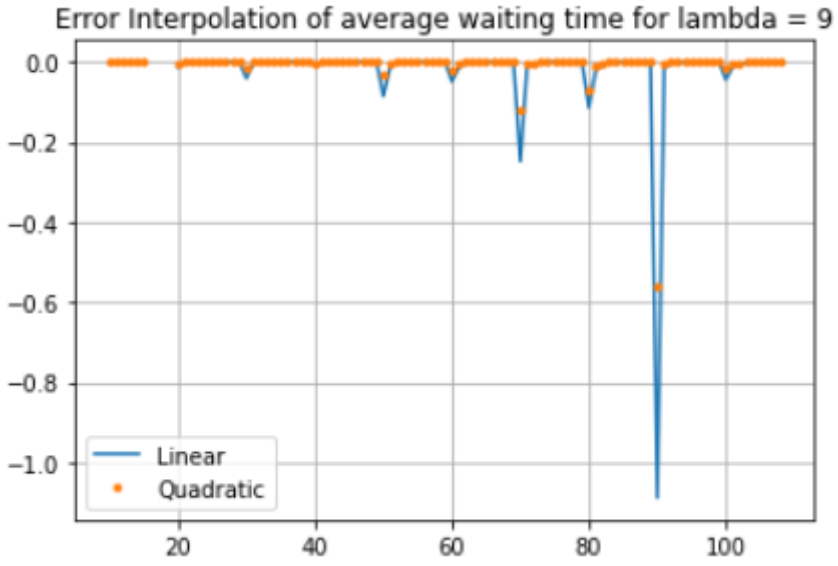

Figure B.2.9: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 9$.

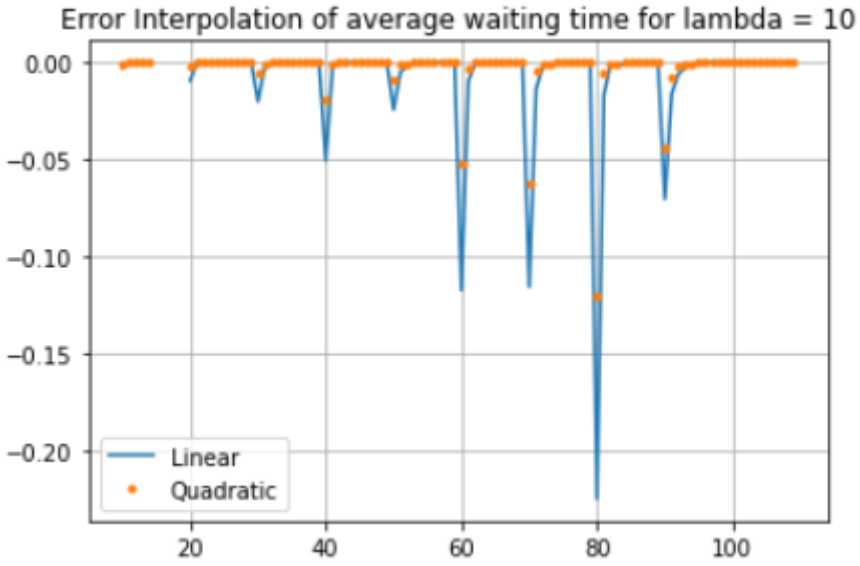

Figure B.2.10: Error between the actual values computed in Python and the interpolation approximation for $\lambda = 10$.

B.3 Staffing level for given service level

In this section, we will show the error of the fractional staffing level between the actual values and the interpolation approximation when a certain target T for the service level is given.

Each graph shows the error for a specific value of the parameter λ considered in the simulation, i.e., $\lambda \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

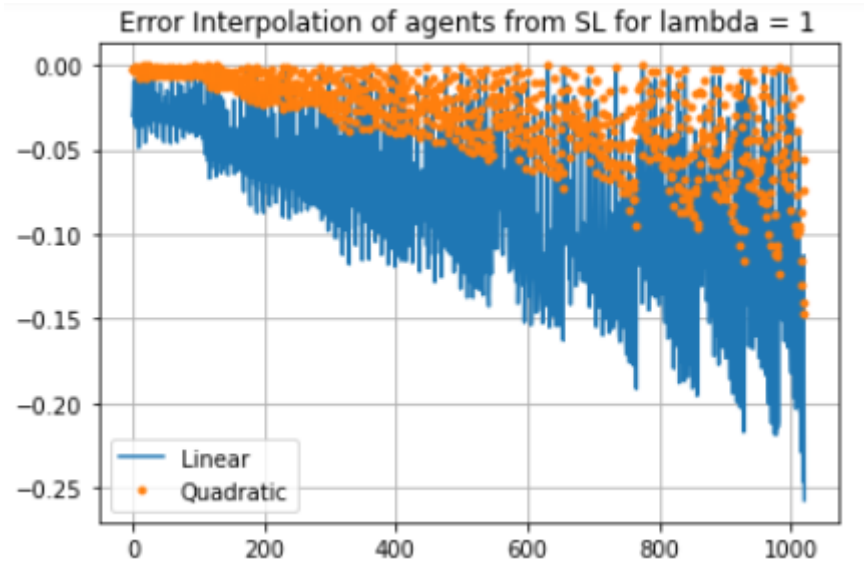

Figure B.3.1: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 1$.

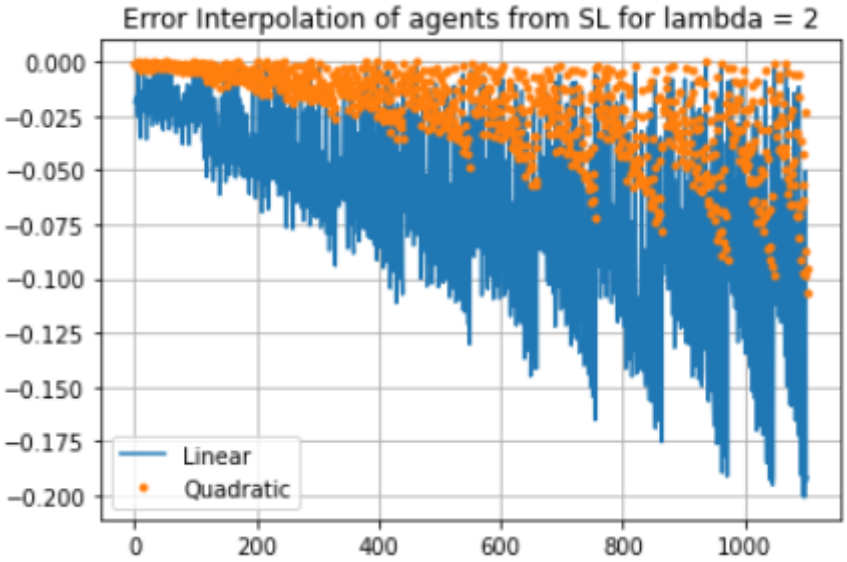

Figure B.3.2: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 2$.

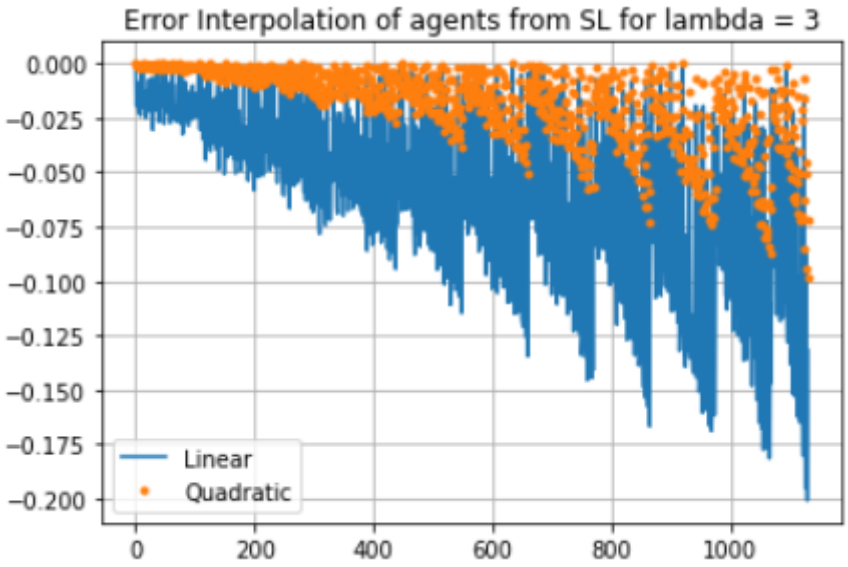

Figure B.3.3: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 3$.

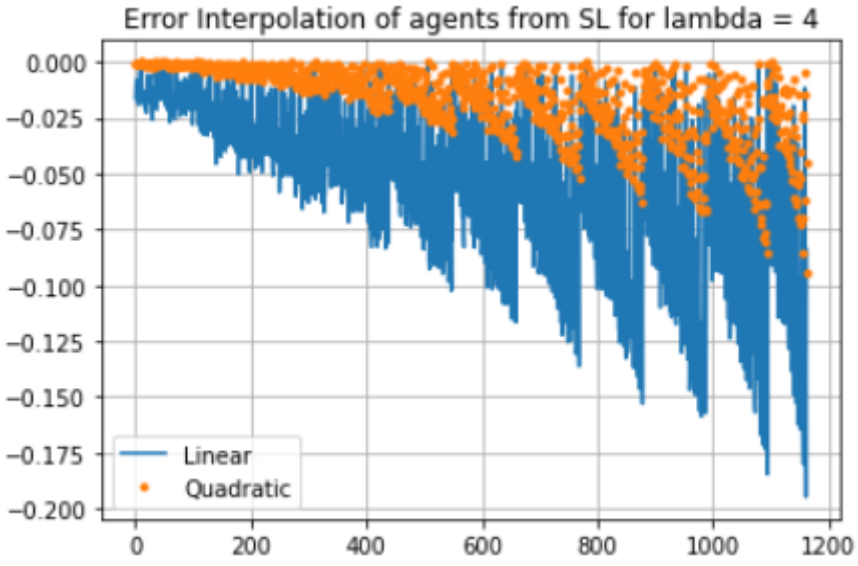

Figure B.3.4: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 4$.

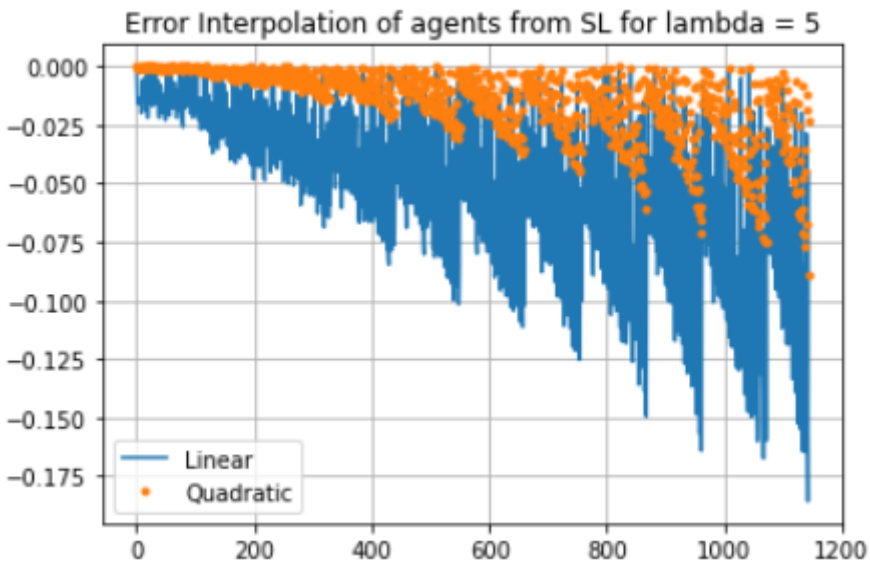

Figure B.3.5: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 5$.

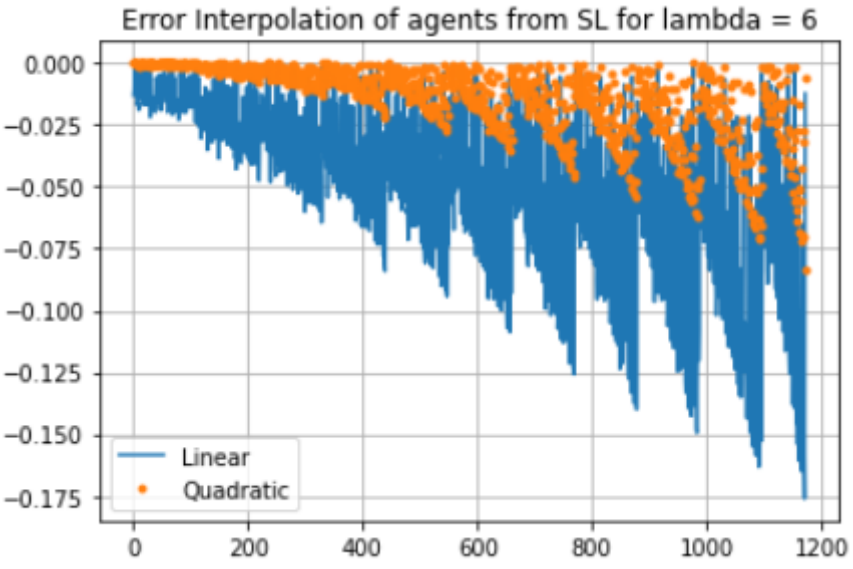

Figure B.3.6: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 6$.

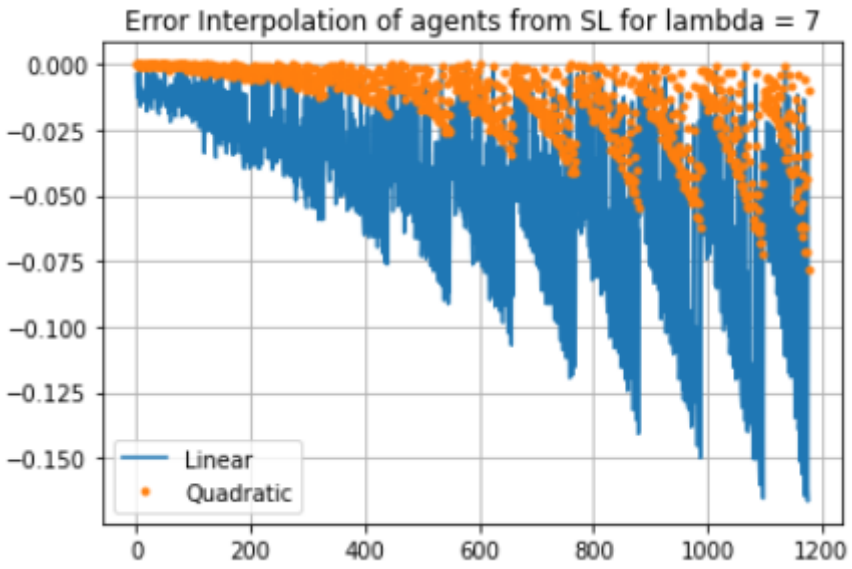

Figure B.3.7: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 7$.

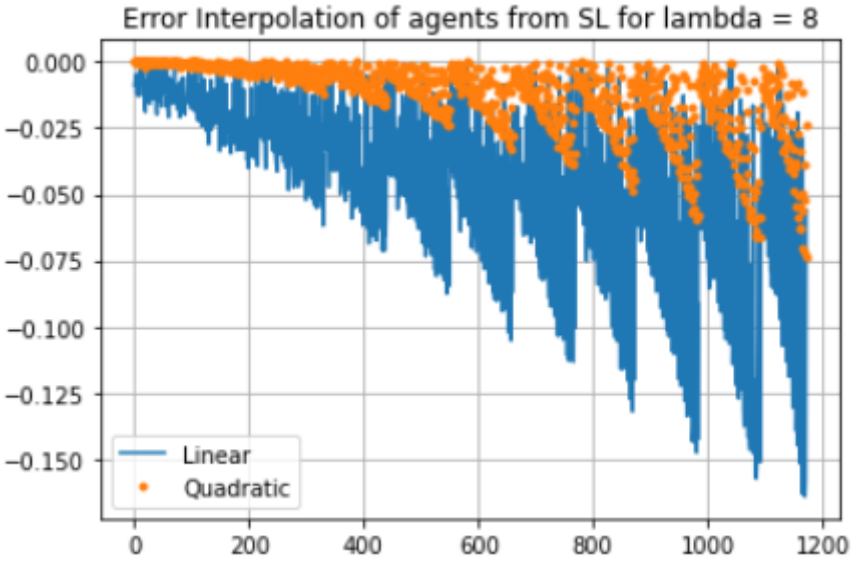

Figure B.3.8: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 8$.

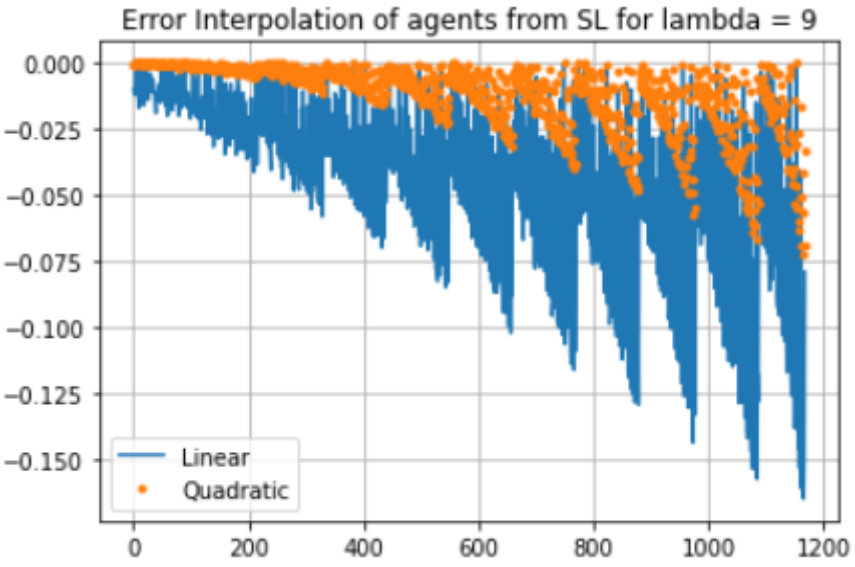

Figure B.3.9: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 9$.

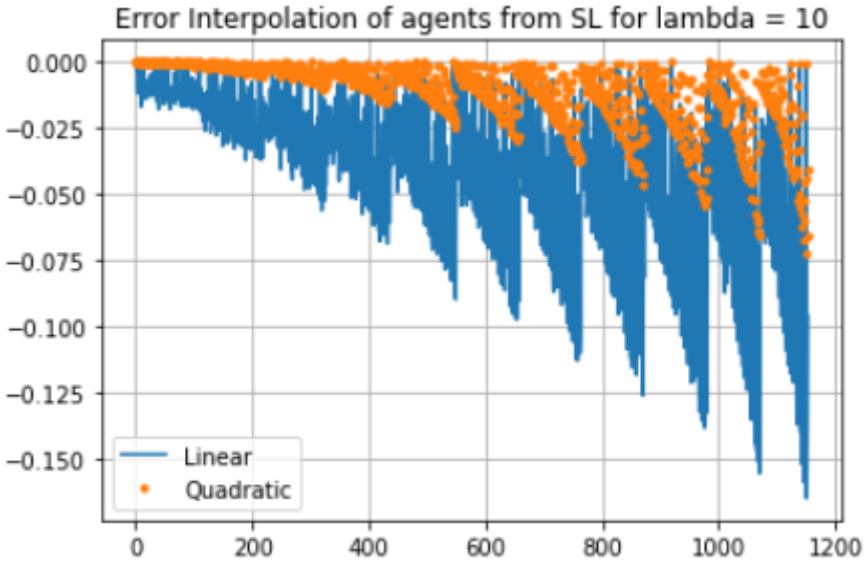

Figure B.3.10: Error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 10$.

B.4 Staffing level for given average waiting time

In this section, we will show the absolute error of the fractional staffing level between the actual values and the interpolation approximation when a certain target T for the average waiting time is given.

Each graph shows the error for a specific value of the parameter λ considered in the simulation, i.e., $\lambda \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

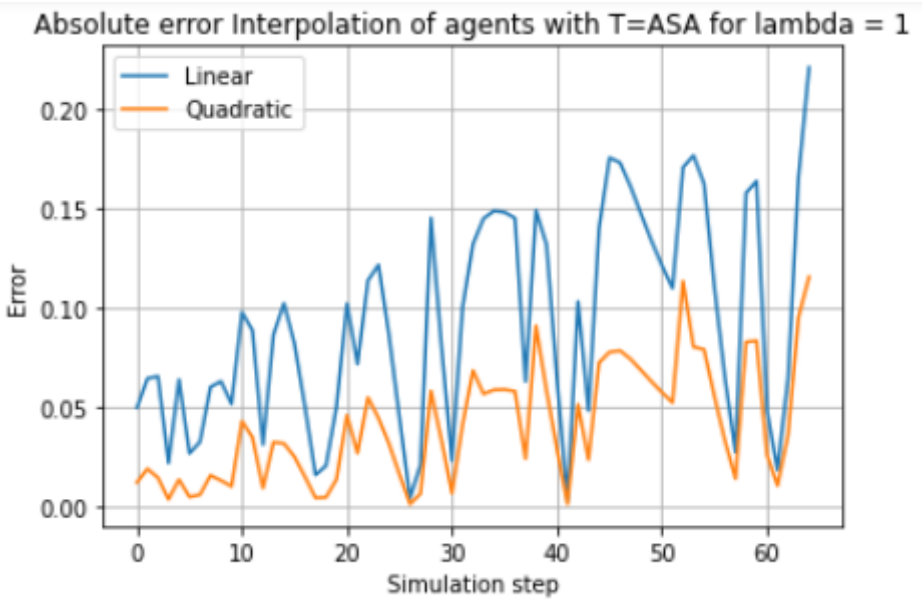

Figure B.4.1: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 1$.

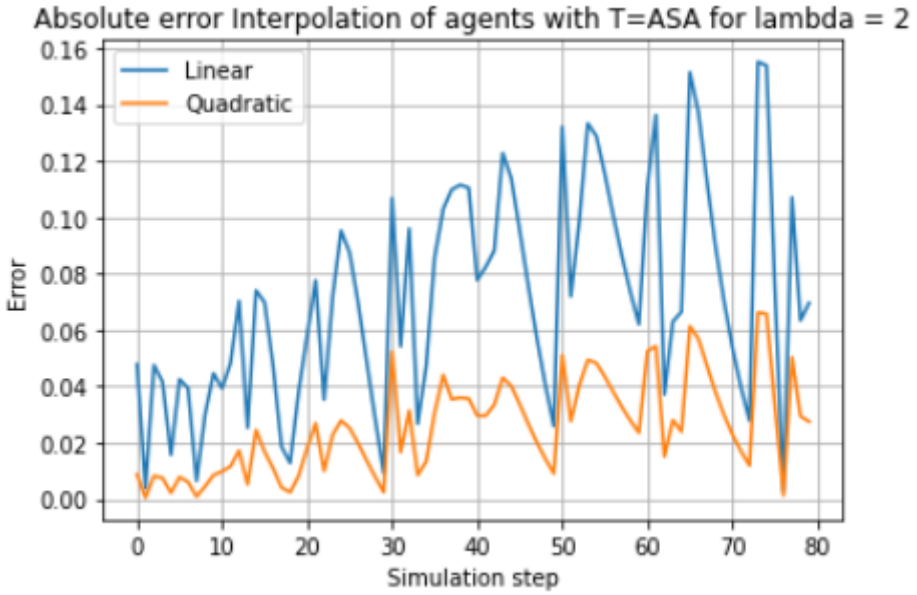

Figure B.4.2: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 2$.

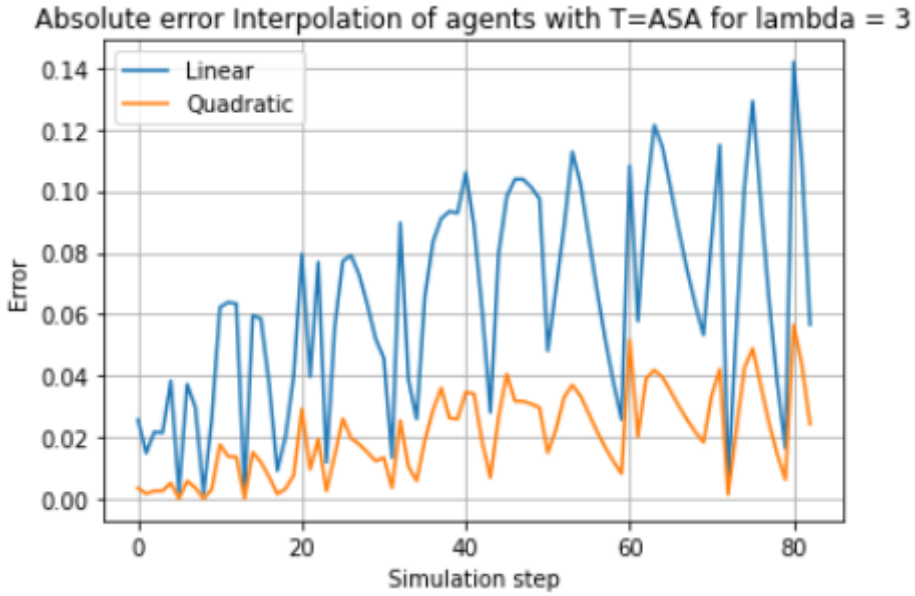

Figure B.4.3: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 3$.

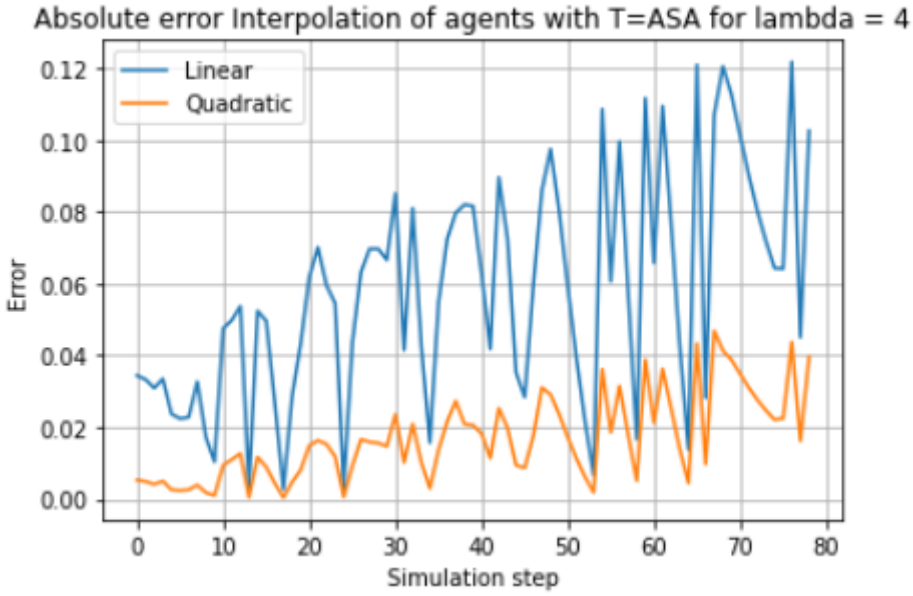

Figure B.4.4: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 4$.

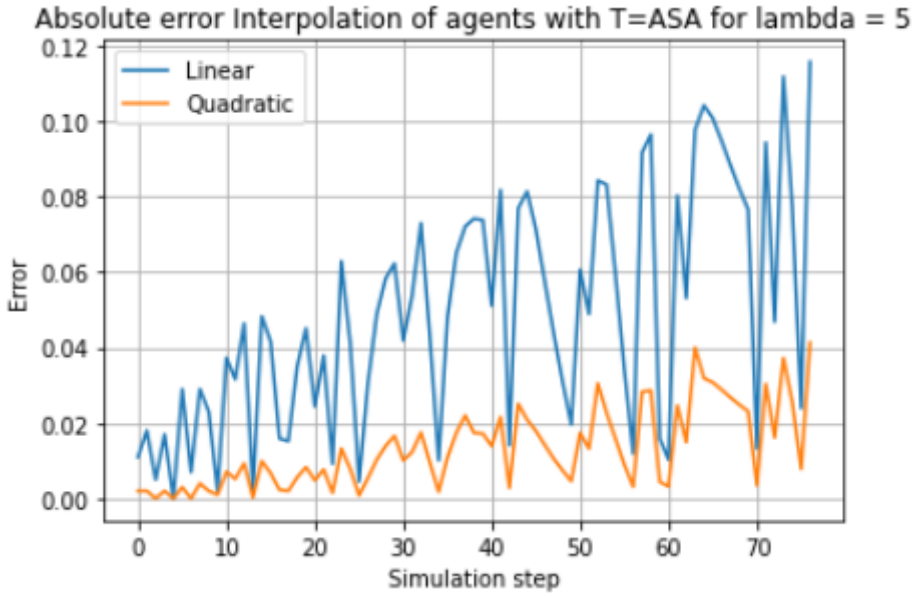

Figure B.4.5: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 5$.

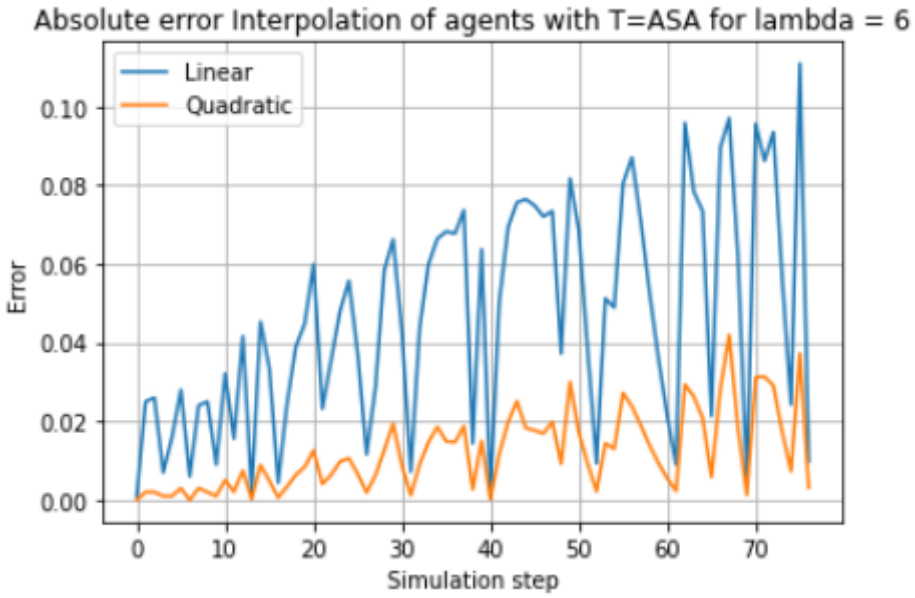

Figure B.4.6: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 6$.

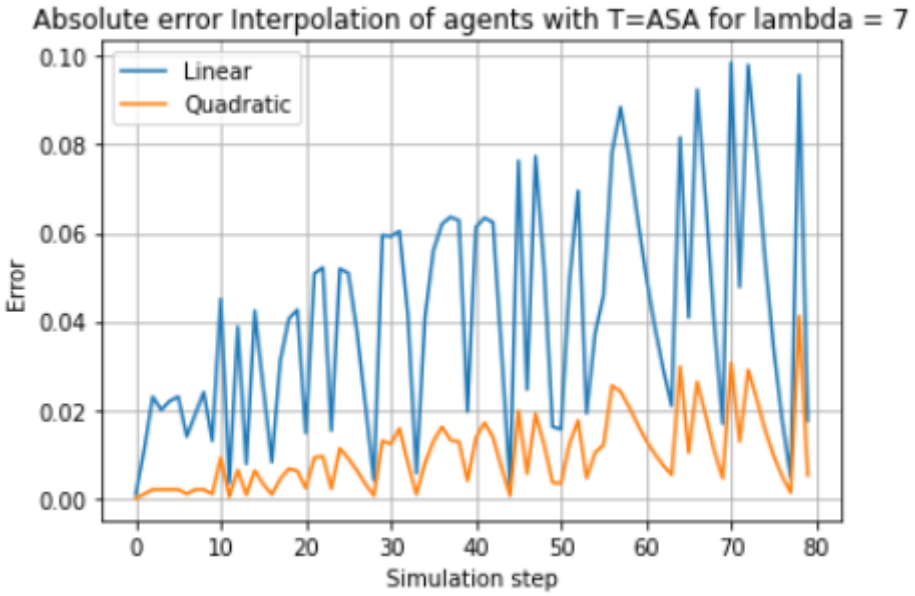

Figure B.4.7: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 7$.

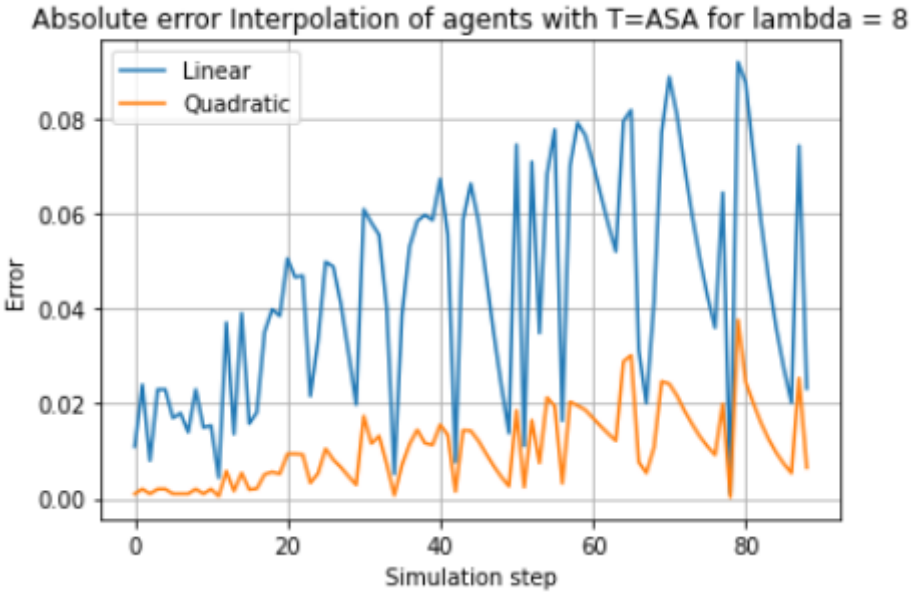

Figure B.4.8: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 8$.

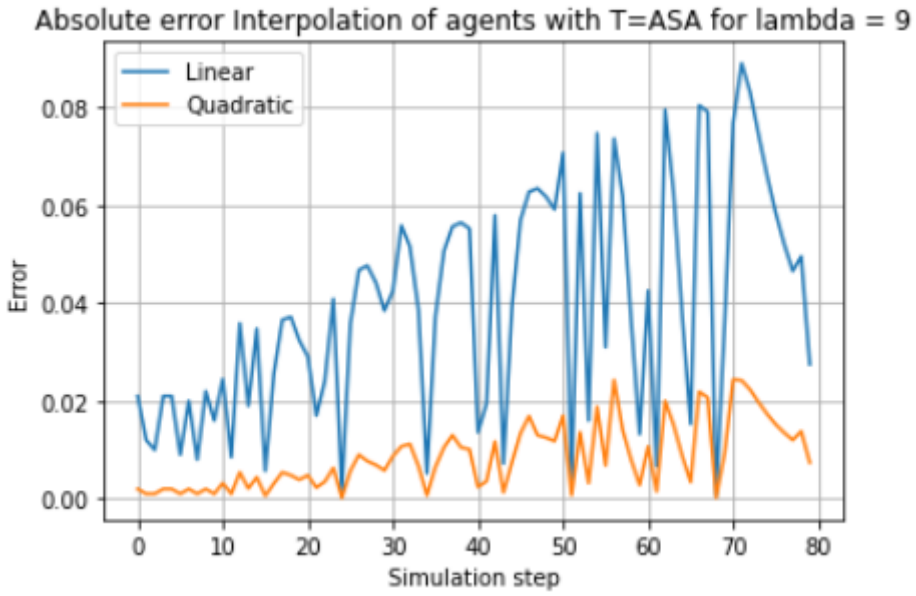

Figure B.4.9: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 9$.

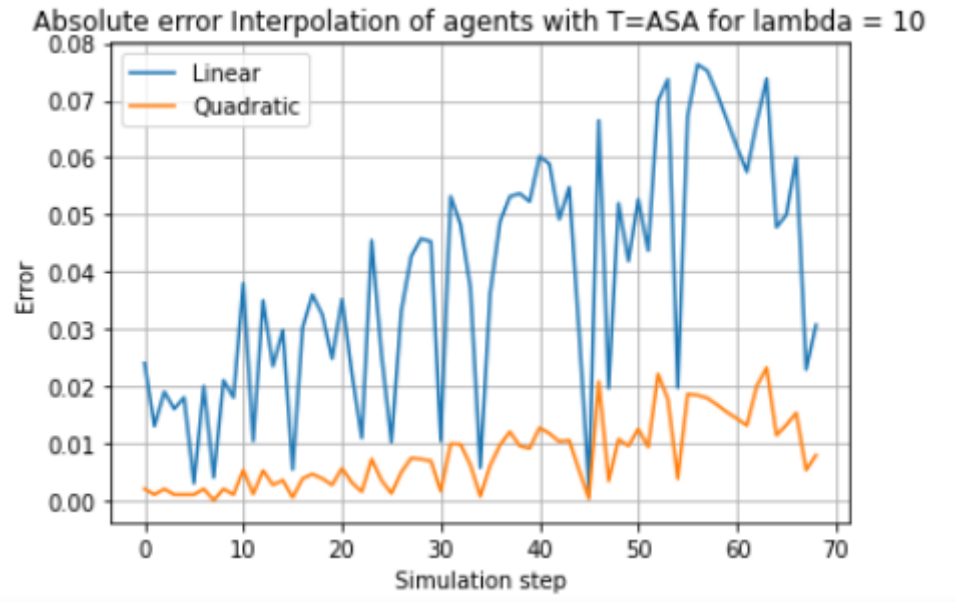

Figure B.4.10: Absolute error between the actual values computed with convergence methods and the interpolation approximation for a given target for $\lambda = 10$.

Bibliography

[1] G.M. Koole. *Optimization of business models*. MG books Amsterdam, 2013.

[2] G.M. Koole and S. Li. A practice-oriented overview of call center workforce planning. *CCmath bv & Vrije Universiteit Amsterdam*, 2022.

[3] G.M. Koole N. Gans and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Institute for Operations Research and the Management Sciences*, 2003.

[4] J. Pichitlamkenb A. Ingolfssonc A. Deslauriersa, P. L'Ecuyera and A.N. Avramidisa. Markov chain models of a telephone call center with call blending. *Computers & Operations Research*, (34), 2007.

[5] D. Levy. *Introduction to Numerical Analysis*. Department of Mathematics and Center for Scientific Computation and Mathematical Modeling (CSCAMM) University of Maryland, 2017.

[6] C.T. Kelley. Iterative methods for linear and nonlinear equations. *Society for Industrial and Applied Mathematics, SIAM*, 1995.

[7] J.F. Hübner M. Dastani and B. Logan. *Programming Multi-Agent Systems*. 10th International Workshop, ProMAS 2012, Valencia, Spain, June 5, 2012.

[8] A. Harel. Sharp and simple bounds for the erlang delay and loss formulae. *Springer Science+Business Media, LLC*, 2009.

[9] A.A. Jagers and E.A. Van Doorn. On the continued erlang loss function. *Departement of Applied Mathematics, Twente University of Technology*, 1986.

[10] A. Mandelbaum. *Call centers*. Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology, 2004.

[11] T. Kadri K. Smaili and S. Kadry. Hypoexponential distribution with different parameters. *online available at http://www.scirp.org/journal/am*, 2013.

[12] N.T. Thomopoulos. *Fundamentals of queueing systems: statistical methods for analyzing queueing models*. Stuart School of Business, Illinois Institute of Technology, 2012.

[13] W. Whitt. Sensitivity of performance in the erlang-a queueing model to changes in the model parameters. *Department of Industrial Engineering and Operations Research, Columbia University*, 2006.

[14] G. Turchetti. *Modelli e Metodi Numerici della Fisica*. Dipartimento di Fisica Astronmia Università di Bologna, 2018.