

Imperial College of Science, Technology and Medicine
Department of Computing

**Analytical Methods and Distribution Fitting
for Response Times
in Markovian Queueing Systems**

Julianna Bor

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy of Imperial College London
October 2025

Declaration of Originality

I declare that this thesis was written by myself, and the presented work is my own, except where stated otherwise.

Copyright Statement

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abstract

Markovian queueing systems are popular in performance modelling due to their analytical tractability and ability to represent diverse real-life systems. In this thesis, we calculate response time distributions — essential for quantile-based international standards and quality of service requirements — for various queueing models and develop an efficient method for model parameterization required by analytical tools calculating response times.

We first focus on the processor-sharing (PS) queueing discipline, known for its analytical complexity. Using the generating function method, we calculate the Laplace-Stieltjes transform of the unconditional response time distribution in an M/M/1-PS queue. We then consider two M/M/1-PS queues in tandem and derive the response time distribution numerically for the first time.

Next, we consider two parallel PS queues with Join-the-Shortest-Queue (JSQ) scheduling. The functional equations obtained contain partial generating functions and their partial derivatives, and therefore cannot be solved by commonly used techniques. We solve these equations numerically and derive the first-ever solution for the response time distribution in two parallel JSQ-PS queues.

Finally, we focus on queues with phase-type (PH) arrival and PH or matrix exponential (ME) service processes, as real-world systems often deviate from exponential assumptions. Accurate response time calculations require properly parameterised distributions, but fitting real-world data to distributions is challenging due to over-parameterization and large equation orders. We introduce a novel fitting method using quasi-birth-death processes parameterised by compositional PH distributions, and a novel ME fitting algorithm. Unlike existing fitting methods which are typically limited to a few states, our methods can fit PH and ME distributions with hundreds of states.

The novel methods proposed in this dissertation advance both the theoretical and practical aspects of performance modelling. Given the importance of response time distributions in performance analysis—a relevance unlikely to diminish—we hope these methods inspire continued innovation in the field.

Acknowledgements

I thank my supervisors, William Knottenbelt and Giuliano Casale for their guidance, encouragement, and invaluable insights.

I am grateful to Peter Harrison for his collaboration, even in retirement, co-authoring several of my papers. I thank Evgenia Smirni and Andreas Stathopoulos for their hospitality during my visit to William and Mary.

I am grateful to my family—Panni, Szása, Anna, Edit, István, and Nagy— for their unwavering emotional support. Most of all, I thank my parents for their support, emotional and financial, which made this PhD possible.

This thesis is dedicated to my niece, Szofi, so she can keep up with her sister, Liza, who already has a PhD dedication.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Background	1
1.2 Motivation and objectives	3
1.3 Contributions	5
1.4 Publications	7
1.5 Thesis outline	9
2 Background	10
2.1 Introduction	10
2.2 Stochastic processes	11
2.2.1 Markov chains	12
2.2.2 Markov processes	13
2.2.3 Markovian queues	15

2.3	Inter-arrival and service distributions	15
2.4	M/M/1 queues alone and in networks	18
2.4.1	Generating functions method	19
2.4.2	A single M/M/1-PS queue	20
2.4.3	Network of M/M/1-PS queues	21
2.5	A single PH/PH/1 queue	22
2.6	Complex analysis	25
2.7	Related work	28
2.7.1	Response time distributions	28
2.7.2	Fitting algorithms	36
3	Response time in a PS queue and two PS queues in tandem	42
3.1	Introduction	42
3.2	M/M/1-PS: exact solution	43
3.2.1	Numerical Results	55
3.3	M/M/1-PS: approximate solution	57
3.3.1	Introductory example	58
3.3.2	Solving the functional equation	61
3.3.3	Numerical results	64
3.4	Tandem PS queues	70
3.4.1	Obtaining the functional equations	70
3.4.2	Solving the functional equations	74

3.4.3	Numerical results	79
3.5	Summary	84
4	Response time in a pair of PS queues with JSQ scheduling	86
4.1	Introduction	86
4.2	Obtaining the functional equations	87
4.2.1	Generating functions of conditional response time distributions	88
4.2.2	LST of the unconditional response time distribution	93
4.3	Solving the functional equations	95
4.3.1	Conditional response times	96
4.3.2	Unconditional response time	98
4.3.3	Moments	99
4.4	Evaluation	101
4.4.1	Comparison with simulation	101
4.4.2	Static scheduling models	104
4.4.3	Finding the optimal routing probabilities	105
4.4.4	Accuracy and cost	107
4.5	Optimisations	108
4.5.1	Optimisation I: inversion of operator order	108
4.5.2	Optimisation II: simplification of the partialization operator	111
4.5.3	Numerical experiments	113
4.6	Summary	115

5	A Compositional PH fitting algorithm	117
5.1	Introduction	117
5.2	Motivating example	119
5.3	Obtaining the queue-length distribution	121
5.3.1	Introducing component queues	121
5.3.2	Approximating the matrix $\mathbf{G}$	122
5.3.3	Approximating the matrix $\mathbf{R}$	129
5.3.4	Approximating the queue-length probabilities	132
5.4	Fitting algorithm	135
5.4.1	Decision variables	136
5.4.2	Outline of the fitting algorithm	136
5.4.3	Using a gamma distribution as $\mathbf{N}$	139
5.4.4	Reducing the size of the fitted distribution	140
5.4.5	Time and space complexity analysis	141
5.4.6	Details of the solver used for the optimisation problem	142
5.5	Numerical results	142
5.5.1	Evaluating Algorithm 1	143
5.5.2	Evaluating the fitting algorithm	144
5.5.3	Response time distributions	146
5.5.4	Advantages and limitations of the proposed algorithm	149
5.6	A compositional ME fitting algorithm	149

5.6.1	Mixture of ME distributions	150
5.6.2	Fitting algorithm	151
5.6.3	Numerical tests	152
5.7	Summary	155
6	Conclusion	158
6.1	Summary of thesis achievements	158
6.2	Future work	163
	Bibliography	164
	Appendices	178
A.1	Chapter 3	178
A.1.1	Proofs	178
A.1.2	Tables and figures	180
A.2	Chapter 5	182
A.2.1	Tables and figures	182
A.2.2	Fitted distribution — Radius Authentication trace	183

List of Tables

3.1	LST in the fourth quadrant in an M/M/1-PS queue with parameters $\lambda = 1$ and $\mu = 3$	57
3.2	LST in the fourth quadrant in an M/M/1-PS queue with parameters $\lambda = 9$ and $\mu = 10$	58
3.3	Numerical results for solving the introductory example $f'(z) = 2z$ with boundary condition $f(0) = 0$ using $n = 8$ discretisation points around the unit circle. . . .	61
3.4	The parameters for the tested scenarios during the numerical comparison	65
3.5	The maximum errors calculated for the test scenarios at $s = 1$ and $s = 10 - 30i$	65
3.6	Comparing the estimates for the LST at $s = 1$ obtained by function G and H . The exact results are calculated by the no-conditioning method.	66
3.7	Comparing the estimates for the LST at $s = 10 - 30i$ obtained by function G and H . The exact results are calculated by the no-conditioning method.	66
3.8	Response time density in an M/M/1-PS queue at selected inputs under light load. . . .	67
3.9	Response time density in an M/M/1-PS queue at selected inputs under medium load.	68
3.10	Response time density in an M/M/1-PS queue at selected inputs under high load. . . .	68

3.11 First four moments of the response time density in an M/M/1-PS queue under light load.	69
3.12 First four moments of the response time density in an M/M/1-PS queue under medium load.	69
3.13 First four moments of the response time density in an M/M/1-PS queue under high load.	69
3.14 Matrices representing operations in two dimensions.	75
3.15 Response time moments for two PS queues in tandem with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$	81
3.16 Response time moments for two PS queues in tandem with parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$	82
3.17 Response time moments for two PS queues in tandem with parameters $\lambda = 2$, $\mu_1 = 2.4$, $\mu_2 = 2.6$	83
3.18 Comparing response time moments for a PS tandem with rates $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$; its order-reversed version with $\mu_1 = 3$, $\mu_2 = 2$; and the convolution of two independent PS queues with $\lambda_1 = 1$, $\mu_1 = 2$ and $\lambda_2 = 1$, $\mu_2 = 3$	83
3.19 Comparing response time moments for a PS tandem with rates $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$; its order-reversed version with $\mu_1 = 1.5$, $\mu_2 = 1.4$; and the convolution of two independent PS queues with $\lambda_1 = 1$, $\mu_1 = 1.4$ and $\lambda_2 = 1$, $\mu_2 = 1.5$	84
3.20 Comparing response time moments for a PS tandem with rates $\lambda = 2$, $\mu_1 = 2.4$, $\mu_2 = 2.6$; its order-reversed version with $\mu_1 = 2.6$, $\mu_2 = 2.4$; and the convolution of two independent PS queues with $\lambda_1 = 2$, $\mu_1 = 2.4$ and $\lambda_2 = 2$, $\mu_2 = 2.6$	84
3.21 Comparing response time moments in PS-PS, PS-FCFS and FCFS-FCFS tandems with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$	85

3.22	Comparing response time moments in PS-PS, PS-FCFS and FCFS-FCFS tandems with parameters $\lambda = 1, \mu_1 = 1.4, \mu_2 = 1.5$	85
4.1	Matrices representing operations in two dimensions.	96
4.2	Matrices representing the operations corresponding to the related analytic functions of Definition 4.1.	96
4.3	Response time moments for two JSQ-PS queues with parameters $\lambda = 1, \mu_1 = 1, \mu_2 = 2, a_1 = 1, a_2 = 0$	101
4.4	Response time moments for two JSQ-PS queues with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$	103
4.5	Response time moments for two JSQ-PS queues with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$	104
4.6	Comparison of response time moments for JSQ/PS, JSQ/FCFS, static load balanced with PS and static load balanced with FCFS. The parameters are $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$	105
4.7	Moments as the probability, a_1 , of joining queue 1 at equal queue lengths varies; The parameters are $\lambda = 2, \mu_1 = 1, \mu_2 = 3$	106
4.8	Moments as the probability, a_1 , of joining queue 1 at equal queue lengths varies. The parameters are $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$	106
4.9	Comparison of the optimised methods with the original algorithm and simulation for calculating response time moments (model parameters: $\lambda = 2, \mu_1 = 1, \mu_2 = 3$ and $a_1 = 0.5$.)	114
4.10	Comparison of the optimised methods with the original algorithm and simulation for calculating response time moments (model parameters: $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$ and $a_1 = 0.5$.)	115

4.11	Comparison of the optimised methods with the original algorithm and simulation for calculating response time moments (model parameters: $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8$ and $a_1 = 0.5$.)	115
5.1	Summary of the main notation	119
5.2	Comparing a cyclic reduction method to the proposed algorithm for inputs with large SCV values. (ARV=Erlang-5 with mean of 2; $S = N \otimes H_1 \otimes \dots \otimes H_J$ where $N = \text{APH}(1, 10)$, $H_i = \text{HE}(1, 1.1)$)	143
5.3	Comparing a cyclic reduction method to the proposed algorithm for inputs with small SCV values. (ARV=Erlang-5 with mean of 2; $S = N \otimes H_1 \otimes \dots \otimes H_J$, $N=\text{APH}(1, 0.1)$, $H_i = \text{HE}(1, 1.1)$)	144
5.4	Log-moments of BCAug89 trace and ME distributions (fitted with and without the Mellin transform).	153
1	Matrices representing operations in the summands of the functional equation for G : G itself; multiplication by a known fixed function $a(z)$; differentiation; and evaluation at a fixed internal point w	181
2	Kolmogorov Smirnov distance for the CDF between the fitted distributions and the trace	182
3	KS distance for the response time CDF between the fitted distributions and the trace ($\rho = 0.25$)	182
4	KS distance for the response time CDF between the fitted distributions and the trace ($\rho = 0.5$)	182

List of Figures

3.1	Compact set B containing $\text{Im}(f_n) \subseteq B$ when n is "large"	51
3.2	Compact sets $B1 : \text{Im}(f_n) \subseteq B1$ and $B2 : \text{Im}(g_n) \subseteq B2$	52
3.3	LST along the positive reals in an M/M/1-PS queue with parameters $\lambda = 1, \mu = 3$ (Light Load) and $\lambda = 9, \mu = 10$ (Heavy Load).	56
3.4	LST in the first quadrant in an M/M/1-PS queue with parameters $\lambda = 1$ and $\mu = 3$	56
3.5	LST in the first quadrant in an M/M/1-PS queue with parameters $\lambda = 9$ and $\mu = 10$	57
3.6	Density function of the response time in an M/M/1-PS queue with parameters $\lambda = 1, \mu = 3$ (Light Load) and $\lambda = 9, \mu = 10$ (Heavy Load).	58
3.7	Steps to approximate $f(z)$ from 8 known values around the unit circle.	59
3.8	Response time density in an M/M/1 PS queue with parameters $\lambda = 2, \mu = 3$ (Medium Load) and $\lambda = 9, \mu = 10$ (Heavy Load).	67
3.9	Two PS queues in tandem with arrival rate λ and service rates μ_1, μ_2	70
3.10	Response time density for a two-server PS tandem network with parameters $\lambda = 1, \mu_1 = 2, \mu_2 = 3$	79

3.11	Response time density for a two-server PS tandem network with parameters $\lambda = 1, \mu_1 = 1.4, \mu_2 = 1.5$.	80
3.12	Response time density for a two-server PS tandem network with parameters $\lambda = 2, \mu_1 = 2.4, \mu_2 = 2.6$.	82
4.1	Two JSQ-PS queues in parallel where the arriving task joins the shorter queue. The arrival rate is λ , the service rates are μ_1 and μ_2 ,	88
4.2	Response time density for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 1, \mu_2 = 2, a_1 = 1, a_2 = 0$; comparison with simulation.	102
4.3	Response time density for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$; comparison with simulation.	103
4.4	Response time density for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$; comparison with simulation.	104
4.5	The contour of integration (in blue) is the distorted circle going around y_0 on the inner side, and the area of convergence (in grey) is the punctured annulus with inner radius r^2 and outer radius 1.	110
5.1	Gamma distribution with $SCV = 0.074$ approximated by APH distributions of increasing size.	139
5.2	Queueing results at 25% utilisation when using KPC-QBD and exact QBD methods to fit queueing probabilities.	145
5.3	Queueing results at 50% utilisation when using KPC-QBD and exact QBD methods to fit queueing probabilities.	145
5.4	Empirical CDF of traces and the corresponding fitted distributions.	146
5.5	Buildserver00 trace CDF of response time, KPC-QBD and benchmark methods.	147

5.6	Radius Authentication trace CDF of response time, KPC-QBD and benchmark methods.	147
5.7	Buildserver00 response time CCDF at 25% utilisation, comparing simulation to reduction.	148
5.8	Radius Authentication response time CCDF at 25% utilisation, comparing simulation to reduction.	148
5.9	ECDF of the BCAug89 trace and CDF of the fitted distributions.	152
5.10	ECDF of the BuildServer00 trace and CDF of the fitted distributions.	153
5.11	Comparing the CCDF of the BCAug89 trace to the ECDF when fitting with and without the Mellin transform.	155
5.12	Steady state queueing probabilities for BCAug89 trace at 25% and 50% utilisation levels.	155
5.13	Steady state queueing probabilities for BuildServer00 trace at 25% and 50% utilisation levels.	156
5.14	Response time CDF for BCAug89 trace at 25% and 50% utilisation levels.	156
5.15	Response time CDF for BuildServer00 trace at 25% and 50% utilisation levels.	157
1	Compact sets $B1 : \text{Im}(f_n) \subseteq B1$ and $B2 : \text{Im}(g_n) \subseteq B2$	180

Chapter 1

Introduction

1.1 Background

Performance modelling is an important tool in software development, system design and capacity planning, network optimization, and operational decision-making. The benefits of performance modelling are, for example, identifying potential performance issues [BWB⁺19], cost reduction through resource optimization [PS03, HP18], improving user experience through better system responsiveness [GAGV15] and facilitating reliability and fault tolerance [TGL24]. There are three main branches of performance modelling: analytical models, simulation-based models and machine learning (ML) models [BGDMT06, Alp21]. Each of these offers distinct advantages and faces unique limitations, making advancements in all three areas essential. Analytical models use mathematical frameworks for predicting system behaviour. They usually have low computational costs, but often require extensive assumptions on certain aspects of the underlying system to fit the mathematical framework. Simulation-based models work by generating events and observing their effects. These models do not require heavy assumptions and can therefore model any system; however, they are often resource-intensive and time-consuming to run. Machine learning models use statistical algorithms to learn from data directly and make predictions about unseen data. These models can identify patterns and relationships, that other types of models might fail to capture. A major drawback of ML models is that sufficient train-

ing of an ML algorithm requires large amounts of high-quality data from a real system which is often unavailable. Additionally, most machine learning models are considered black box models, making it difficult to understand why they produce certain results and, therefore, challenging to draw meaningful insights from them, or explain why different models trained on the same data may produce different results.

In this dissertation, we focus on queueing systems, a subset of analytical models. The underlying mathematical framework is known as queueing theory, which studies a specific class of stochastic models [HP92]. Stochastic processes describe systems whose state evolves over time, and where changes from one state to another are not deterministic. They are used widely in computational finance, communication and computer networks. Within stochastic modelling, queueing theory focuses on systems where there is contention for one or more resources. There are numerous applications of queueing models, such as the analysis of supply chains that carry perishable items [GW19], the evaluation of an IaaS cloud system [AER⁺19] or the modelling of user-level mobility in a campus wireless network [CKT12]. A basic queueing model consists of a waiting room, a corresponding scheduling discipline that specifies the order of service, and arrival and service processes. Under certain assumptions on the arrival and service processes, the resulting queueing model is Markovian. Furthermore, queues can be bounded, that is, having a waiting room of fixed capacity, where arrivals are ignored when the queue is full, or unbounded, meaning arrivals can always join the queue. In this work, we consider only unbounded queues.

As previously mentioned, performance modelling can be used to improve many aspects of real-world systems. For instance, queueing models have been applied to improve both responsiveness and fault tolerance in a network by using replicas [HQ13]. The reliability of communication systems is examined in [RPT⁺15], while the security of public clouds is addressed in [EM17]. When considering user experience in a resource contention context, responsiveness is one of the most critical factors; specifically, how long it takes to serve a customer or complete a task. This has been a central focus of queueing theory, with the corresponding performance measure referred to as response time or sojourn time. Applications range from minimising the waiting time in an emergency department [HKH⁺16] to optimising the scheduling of police cars [KRCW75]. While the mean response time is the most straightforwardly obtainable indicator, there are several

scenarios where simply looking at the average response times alone does not provide enough insight into the behaviour of the system. For example, a system with a mean response time of one minute and a low standard deviation is more likely to appeal to users than another system, where the mean response time is ten seconds, but occasionally it exceeds an hour. Additionally, service level agreements (SLAs) often use quantiles to specify performance requirements, e.g. the response time needs to be under ten seconds 90% of the time. Consequently, obtaining the probability distribution of response time is important for evaluating system performance and it is the main focus of this dissertation.

1.2 Motivation and objectives

Despite its long history, queueing theory continues to face several challenges. One such challenge arises when considering a specific queueing discipline known as processor-sharing (PS). The queueing discipline determines the order in which the currently waiting customers are served. In a PS queue, tasks are served concurrently, with each task receiving a $1/n$ th share of the total serving capacity, given there are n customers in the queue. Compared to PS queues, queues operating under first-come-first-served (FCFS) or last-come-first-served (LCFS) discipline are much easier to handle analytically. Even if the queue itself has exponential inter-arrival and service times — denoted as $M/M/1$ queue using Kendall's notation — solving for the response time density becomes significantly more complex under the PS discipline. Despite these analytical difficulties, the PS discipline is often used in applications. However, in comparison to other disciplines like FCFS or LCFS, our understanding of their performance properties is less developed, especially concerning response time distributions. Existing results are limited, particularly in the context of networks of queues, even in the simplest case of two PS queues in tandem. Our first research objective (RO1) is therefore defined as follows.

RO1: Explore novel approaches for obtaining response time distributions in PS queues.

Another relevant example of a model where PS scheduling introduces significant challenges is as follows. Imagine a system with several parallel queues, where arriving jobs join one of these

queues. Join-the-Shortest-Queue (JSQ) is a scheduling policy that assigns arriving jobs to the shortest of a number of parallel queues and breaks ties by assigning jobs randomly with a given probability. JSQ—also called Least-Connection scheduling in a load balancing context—is the policy of choice for many network providers, cloud servers, and traffic management systems, where individual queues are served under processor sharing (PS) queueing discipline. For example, JSQ is offered by Cisco IOS Server [WC11], Azure Application Gateway [WAZ23], Kubernetes [WK23], and AWS Application Load Balancer [WA19]. Despite the ubiquity of JSQ scheduling, there is still a lot unknown regarding its quantitative behaviour. In particular, the combination of JSQ scheduling and PS discipline at the individual queues remains an unsolved problem, even in the simplest case of just two parallel queues.

RO2: Obtain the response time distribution in a pair of PS queues with JSQ scheduling.

Exact solutions to queueing models regularly rely on the assumption that the inter-arrival and service times are exponentially distributed. However, real-world systems often do not follow exponential distributions. One possible solution to this problem is to consider queues with phase-type (PH) arrivals and service processes [LR99], or their generalised version known as matrix-exponential (ME) distributions [BN12]. The main advantage of PH distributions, and consequently ME distributions, is that the set of PH distributions is dense in the set of non-negative distributions; therefore, any non-negative distribution can be arbitrarily well-approximated by a PH distribution. There are analytical tools that can calculate the response time for a $PH/PH/1$ or an $ME/ME/1$ queue with given parameters. However, in order to find the correct set of parameters that represent system measurements accurately, the distribution of arrival and service processes needs to be fitted to actual data. Fitting real-world data is difficult in practice due to model over-parameterization [ANO96], the large order of the fitting equations, and the limited understanding of how fitting specific features of a distribution influences the accuracy of performance predictions, such as response time. Consequently, existing fitting methods are often limited to PH and ME distributions with a handful of states due to computational bottlenecks. This is a significant limitation as the number of states is linearly proportional to the degrees of freedom available for fitting [CS12]. Furthermore, the use of ME

distributions in queueing theory is hampered by the difficulty of checking their feasibility. As a result, there are very few algorithms available for fitting ME distributions.

RO3: Explore compositional approaches for developing novel PH fitting and ME fitting algorithms.

1.3 Contributions

Motivated by the challenges identified in the previous section, the contributions of this thesis are summarised as follows.

- To address **RO1**, we calculate the Laplace-Stieltjes transform of the response time distribution using the generating function method in a single PS queue, without conditioning on the service time of the tagged job. (This additional condition on the service time is how the original result was obtained by Coffman in 1970). The functional equation obtained for the generating function of conditional response time transforms is solved in two different ways. The first version derives an exact solution, which was implemented in Mathematica and verified against Coffman's original result. The second approach involves adopting a novel approximation method that transforms the functional equation into a system of linear equations. The results of the approximation are extensively validated by comparing them to the exact solution of the previous section at various stages of the calculations. Additionally, moments are calculated efficiently by an obtained recursive formula. The approximation method demonstrates high accuracy, even under heavy load conditions, making it a promising alternative for analysing response times in PS queues in more complex settings. Motivated by these findings, we derive functional equations for the LST of response time distribution in a tandem network of two PS queues and apply the above-mentioned approximation method to solve the equations, and consequently obtain the response time density by numerical inversion of the LST for the first time. Again, a recursive formula is derived for efficient calculation of the response time moments. The results are validated by comparing them against regenerative simulation and show high accuracy.

- **RO2** is achieved by deriving a numerical solution for the response time distribution in two parallel PS queues with JSQ scheduling for the first time. Using the generating function method, two partial differential equations (PDEs) are obtained corresponding to conditional response times, where the conditioning is on a particular traced task joining the first or the second queue. These PDEs are functional equations that contain partial generating functions and their partial derivatives, and therefore cannot be solved by commonly used techniques. We are able to solve these PDEs numerically with good accuracy and perform the deconditioning with respect to the queue-length probabilities by evaluating a certain complex integral. Numerical results for the density and the first four moments compare well against regenerative simulation.
- To address the PH fitting part of **RO3**, we present a methodology for fitting measured queueing data using quasi-birth-death (QBD) processes parameterized by phase-type (PH) distributions generated through Kronecker product composition. Our methodology can increase fitting accuracy by enabling a search directly over a space of QBD processes. Core to the proposed method is a new result that reduces the solution of a broad class of QBD processes to the analysis of a collection of simpler queueing systems featuring PH distributions of low order. This leads to exact or approximate solutions depending on the chosen arrival process. The proposed method reduces the computational complexity of solving a QBD process from cubic to quadratic for this special class of QBD processes, enabling much more efficient search. Unlike existing fitting methods, which typically focus on arrival and service processes with a handful of states, our methodology can efficiently analyse and fit QBD processes parameterized with PH distributions having thousands of states. Numerical results on real traces indicate that this feature enables accurate QBD process fitting of real-world system data.
- To address the ME fitting part of **RO3**, we propose a novel ME fitting algorithm that produces a valid distribution by construction. The ME distribution used during the fitting is a product of independent random variables that are easy to control in isolation. Consequently, the calculation of the CDF and the Mellin transform factorizes, making it possible to use these measures for the fitting without significant restriction on the distri-

bution order. Trace-driven queueing simulations indicate that the resulting distributions yield highly accurate results.

1.4 Publications

[BH25b] J Bor, Peter G Harrison. **Response time in a tandem network of two processor sharing queues** 21st European Performance Engineering Workshop (EPEW), 2025, *Best presentation award*

This paper provides the first-ever solution for the response time density of two PS queues in tandem. It formulates exact functional equations and solves them using a numerical approximation technique. The method yields highly accurate results when validated against simulation. The technical details are presented in Chapter 3. My contributions to this paper include theoretical analysis, providing rigorous proofs for the necessary lemmas and propositions, numerical validation, and writing.

[BH24] J Bor, Peter G Harrison. **Response time in a pair of processor sharing queues with Join-the-Shortest-Queue scheduling** 32nd International Conference on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2024, *Best paper award*

This paper presents the first-ever solution for the response time density of two parallel PS queues under JSQ scheduling. It derives exact functional equations which are then solved using a numerical approximation technique. The method produces highly accurate results when compared against simulation. The details of this work are included in Chapter 4. I made the following contributions to this paper: theoretical analysis, including providing rigorous proofs for the required lemmas and propositions, numerical validation, and writing.

[BH25a] J Bor, Peter G Harrison. **Response time in a pair of processor sharing queues with Join-the-Shortest-Queue scheduling**, Performance Evaluation, 2025

This is an extended version of [BH24] that was invited to be published in a Special Issue of Performance Evaluation. The extension includes the proofs of the lemmas and propositions

which were left out of the conference version due to space constraints. Furthermore, two optimization techniques have been added that further improve the accuracy of the original algorithm. I contributed to the derivation and proof of the lemmas on which the optimization methods are based, the extensive numerical evaluation of these methods and the writing. The detailed contributions of this work are presented in Chapter 4.

[BCS25] J Bor, G Casale, E Smirni. **The Component Queues Method for Quasi-Birth-Death Process Fitting and Analysis**, under review, 2025

This work develops a novel algorithm for fitting real-world traces to a $PH/PH/1$ queue. It uses a compositional service process based on Kronecker products and derives theorems based on which the solution of the corresponding QBD is reduced from cubic to quadratic. The speed-up allows the fitting of PH service distributions with thousands of states, a development that has not been observed elsewhere. Furthermore, it allows for the fitting to be conducted based on the queue-length traces, directly. Numerical results show the validity of the approach. I was the main contributor to the paper, my contributions include developing the theoretical framework underlying the method, designing and implementing the fitting algorithms, conducting numerical experiments and writing the manuscript. The details of this work are presented in Chapter 5.

[BCK⁺23] J Bor, G Casale, W Knottenbelt, E Smirni, A Stathopoulos. **Fitting with matrix exponential mixtures generated by discrete probabilistic scaling**, ACM SIGMETRICS Performance Evaluation Review, 2023

In this paper, we develop an algorithm that uses the family of matrix exponential distributions for fitting service traces. The novelty of the paper is the guarantee that the method produces a valid distribution, addressing the primary challenge when using ME distributions in a fitting context. It uses a compositional approach and therefore allows fitting without significant restriction on the distribution order. Trace-driven queueing simulations indicate that the resulting distributions yield highly accurate results. I was the main contributor to this paper, developing the theoretical framework, designing and implementing the corresponding algorithms, conduct-

ing extensive evaluation and writing the manuscript. This work is presented in the thesis as part of Chapter 5.

[HB21a] Peter G Harrison, J Bor. **Response Time Distribution in a Tandem Pair of Queues with Batch Processing**, Journal of the ACM (JACM), 2021

This paper presents an exact solution for calculating response time density in a tandem pair of Markovian queues with both batch arrivals and batch departures. My main contribution was a novel theorem that allows the extraction of certain (infinitely many) coefficients from a double sum by evaluating a complex integral, which addresses a critical problem in the paper. The work was carried out prior to the PhD, however, the theorem was later adapted to solve a similar issue in [BH24, BH25a]. The adapted theorem is presented in Chapter 4.

1.5 Thesis outline

The structure of the rest of dissertation is organized as follows:

- Chapter 2 introduces the mathematical background required in the consecutive chapters.
- Chapter 3 presents two novel solutions for the response time density in an M/M/1-PS queue; and obtains the response time density in a tandem network of two PS queues for the first time.
- Chapter 4 obtains the response time of two parallel PS queues with JSQ scheduling, a previously unsolved problem.
- Chapter 5 presents two novel fitting algorithms and the corresponding theoretical frameworks.
- Chapter 6 summarises the contributions of this thesis and outlines possible future directions.

Chapter 2

Background

2.1 Introduction

The focus of this dissertation is on analytical performance models, more specifically, queueing models. Analytical models can be solved by exact or approximate methods, both of which typically involve some level of numerical approximation. An example of an exact solution is deriving a closed-form formula for the Laplace-Stieltjes transform of the response time distribution, which is then inverted numerically to obtain the response time density [HB21b]. Exact and approximate approaches are often combined; for example, when deriving exact equations that are not solvable analytically, numerical approximation techniques can be utilised to find a good approximate solution. While exact solutions are generally preferable, they are only achievable under significant constraints on the type of queueing models considered. An assumption commonly made when attempting to obtain exact solutions is exponential inter-arrival and service times. One advantage of this assumption is that the exponential distribution is memoryless; therefore, conditioning on the current state of the process, the time to the next state transition is always known, it is the minimum of several exponential random variables. Given a large set of users acting independently of one another, say, submitting requests to a server, a model with exponentially distributed inter-arrivals matches real-life observations well. However, inter-arrival times, and especially service times, may follow a non-exponential distribution. Without

heavy restrictions obtaining exact formulas is often not feasible or takes too long to calculate in practice. Consequently, a good approximate solution is equally valuable as it relaxes some of the assumptions and allows for a wider range of systems to be modelled effectively. In this thesis, we explore all three approaches: exact solutions, approximations, and their combinations.

In this chapter, we present an overview of the theoretical foundation required to understand the rest of the thesis. A more detailed source for the results and their corresponding proofs can be found in [HP92, LR99, BC09]. The structure of the chapter is as follows. Section 2.2 summarises the key definitions and theorems related to discrete and continuous-time Markov chains. Section 2.3 introduces the probability distributions used for inter-arrival and service times in the thesis. Section 2.4 introduces the $M/M/1$ queue, outlines the generating functions method used for obtaining the response time distributions in Chapter 3 and 4, and describes the queueing models used in those chapters. Section 2.5 explores the $PH/PH/1$ queue and quasi-birth-death processes, which are central to the analysis in Chapter 5. Section 2.6 compiles the definitions and theorems from complex analysis used throughout the thesis. Finally, Section 2.7.1 and Section 2.7.2 review existing approaches for obtaining response time distributions and distribution fitting, respectively.

2.2 Stochastic processes

Stochastic processes use randomness to model a system whose state evolves over time. Formally, a *stochastic process* is a family of random variables $\{X_t | t \in T\}$, where T is the parameter space often referred to as time, and each X_t takes its values from a set S called the state space.

A stochastic process $\{X_t\}$ has the *Markov property* if $\forall \tau_1 < \dots < \tau_m < t_1 < \dots < t_n \in T$

$$F_{X_{t_1} \dots X_{t_n} | X_{\tau_1} \dots X_{\tau_m}}(x_1, \dots, x_n | y_1, \dots, y_m) = F_{X_{t_1} \dots X_{t_n} | X_{\tau_m}}(x_1, \dots, x_n | y_m)$$

In other words, the Markov property states that the future state of the system only depends on the current state and not on the history of past states.

2.2.1 Markov chains

A discrete-time stochastic process with a countable sample space that has the Markov property is called a *Markov chain*. It is said to be time-homogeneous if given the process is in state i at time t , the probability that it enters state j sometime after that only depends on the elapsed time and not t . A time-homogeneous Markov chain is characterised entirely by its one-step transition probabilities p_{ij} and a probability vector describing the initial probabilities of starting in each state. The transition matrix $\mathbf{P} = [p_{ij}]_{i,j \in S}$ of a *discrete-time Markov chain (DTMC)* is a stochastic matrix, that is $\forall i : \sum_j p_{ij} = 1$. Let $p_{ij}^{(n)}$ denote the n -step transition probabilities and $\mathbf{P}^{(n)}$ the corresponding transition matrix.

The Chapman-Kolmogorov equation states that $\mathbf{P}^{(m+n)} = \mathbf{P}^{(m)}\mathbf{P}^{(n)}$. Therefore, the n -step transition matrix can be calculated from the one-step transition matrix as $\mathbf{P}^{(n)} = \mathbf{P}^n$.

When modelling a real-world system, we are often interested in its behaviour at equilibrium, assuming this exists. The steady-state distribution of a Markov chain is a probability vector $\vec{\pi}$ that satisfies $\vec{\pi}\mathbf{P} = \vec{\pi}$.

The Fundamental Theorem of Markov Chains specifies the conditions under which a Markov chain has a steady state. To state this theorem, we first introduce the following definitions.

A Markov chain is said to be *irreducible* if the probability of getting from any state to any other state in n steps is positive for some n . In other words, it is possible to enter any state in a finite step regardless of the current state.

A Markov chain is said to be *aperiodic* if the m step transition probabilities are positive for any m with finitely many exceptions.

Finally, a Markov chain is *positive recurrent* if the process returns to any state infinitely often with a finite expected recurrence time.

Theorem 2.1 (Fundamental Theorem of Markov Chains). *In an irreducible, aperiodic Markov chain,*

1. $p_{ij}^{(n)} \rightarrow \frac{1}{E(T_{jj})}$ as $n \rightarrow \infty$
2. The chain is positive recurrent $\Leftrightarrow$ There is a unique stationary distribution $\vec{\pi}$, such that $\vec{\pi} = \vec{\pi} \mathbf{P}$ with $\pi_j = \frac{1}{E(T_{jj})}$

where T_{jj} is the time of the first return to state j and $E(T_{jj})$ is its expected value.

Therefore, in an irreducible, aperiodic, and positive recurrent Markov chain, the steady-state distribution is the same as the limiting distribution. In practice, we can find a solution to the equation $\vec{\pi} = \vec{\pi} \mathbf{P}$ which always exists and then normalise it. The elements of $\vec{\pi}$ summing to 1 is a necessary and sufficient condition for $\vec{\pi}$ to be a distribution. Consequently, using the theorem above, a non-zero normalising constant is a necessary condition for the chain to be positive-recurrent.

2.2.2 Markov processes

A stochastic process with continuous parameter space and countable state space which has the Markov property is called a *Markov process* or a *continuous-time Markov chain (CTMC)*. We use the same notation as for Markov chains, we further assume the transition matrix $\mathbf{P}(t) = [p_{ij}(t)]_{i,j \in S}$ satisfies the following conditions,

1. $\forall t \in [0, \infty) : \mathbf{P}(t)$ is stochastic
2. $\mathbf{P}(t)$ is right continuous at $t = 0$
3. $\mathbf{P}(s+t) = \mathbf{P}(s)\mathbf{P}(t)$

The third condition is the continuous time equivalent of the Chapman-Kolmogorov equations. It is also a consequence of the Markov property.

Assume the process enters the state i at time t and the next transition is at $t + T$, T is called the *state holding time* of state i . T has an exponential distribution, with parameter q_i which is also referred to as the *transition rate* from state i . Depending on the value of q_i , a state i is said to be,

1. *stable* if $0 \leq q_i < \infty$
2. *instantaneous* if $q_i = \infty$

3. *absorbing* if $q_i = 0$

Let $\mathbf{Q} = [q_{ij}]_{i,j \in S}$ be a matrix for which $\mathbf{P}(h) = \mathbf{I} + \mathbf{Q}h + o(h)$ and therefore $\mathbf{Q} = \frac{\partial}{\partial t}\mathbf{P}(0)$. Such $\mathbf{Q}$ always exists due to the Chapman-Kolmogorov equation. It can be shown that

$$\begin{aligned} \forall i \neq j : P(X_{t+h} = j | X_t = i) &= q_{ij}h + o(h) \\ i = j : -q_{ii} &= \sum_{j \in S} q_{ij} = q_i \end{aligned}$$

In other words, q_{ij} is the *instantaneous transition rate* from state i to state j , and summing for all j we get the instantaneous transition rate from state i .

$\mathbf{Q}$ is also called the *generator matrix* of the Markov process since it determines the evolution of the process entirely through the following equations:

1. *Forward Kolmogorov equation*: $\exists M < \infty \forall i : q_i < M \Rightarrow \frac{\partial}{\partial t}\mathbf{P}(t) = \mathbf{P}(t)\mathbf{Q}$
2. *Backward Kolmogorov equation*: $\forall i : q_i < \infty \Rightarrow \frac{\partial}{\partial t}\mathbf{P}(t) = \mathbf{Q}\mathbf{P}(t)$

Due to the above equation, the transition matrix $\mathbf{P}(t)$ of a CTMC can be calculated from the generator matrix $\mathbf{Q}$ as $\mathbf{P}(t) = e^{\mathbf{Q}t}$, where $\sum_{k=0}^{\infty} \frac{\mathbf{Q}^k t^k}{k!}$ is a matrix exponential.

Similar to the Fundamental Theorem of Markov chains, we have the following result for the steady-state behaviour of a Markov process.

Theorem 2.2. *An irreducible Markov process with generator matrix $\mathbf{Q}$ is positive recurrent if and only if $\vec{\pi}\mathbf{Q} = \vec{0}$ has a non-zero solution.*

It is often more convenient to work with the discrete-time process we get by recording the state right after each transition. The precise definition is as follows:

Let $\{X_t | t \geq 0\}$ be a Markov process, and let $\tau_1, \tau_2, \dots$ denote the time instants at which the state changes. Then the chain defined as $\{Z_n | n = 0, 1, \dots\}$ where $Z_0 = X_0$ and $Z_n = X_{\tau_n^+}$ has the Markov property and it is called the *embedded Markov chain* (EMC).

A Markov process is said to be irreducible, or positive recurrent if the corresponding EMC is. There is a direct relationship between the steady-state distribution of the CTMC and its

EMC. Denoting the steady-state probability vector of the EMC as $\vec{\gamma}$, we have $\pi_i = \frac{\gamma_i q_i^{-1}}{\sum_{j \in S} \gamma_j q_j^{-1}}$. Therefore solving the EMC, immediately gives the solution to the CTMC.

2.2.3 Markovian queues

A queue consists of an arrival process, a service process and a waiting room, also referred to as the queue, where the customers wait to be served by one of the servers assigned to the queue. Queues are classified according to Kendall's notation: $A/S/m$, where A stands for the arrival process, S is the service time distribution, and m denotes the number of servers serving the queue. The length of the queue over time can be modelled as a stochastic process. Depending on the type of arrival and service processes, this stochastic process may be a CTMC, in which case the queue is said to be Markovian.

2.3 Inter-arrival and service distributions

In this section, we define the distributions used in later chapters to characterise the inter-arrival and service times in the queueing models under study. We also highlight their most relevant properties and characteristics.

Exponential distribution

A random variable X is exponentially distributed with rate λ , that is $X \sim \text{Exp}(\lambda)$, if the density function $f(t)$ is

$$f(t) = \lambda e^{-\lambda t} \quad t \geq 0$$

The mean can be calculated as $E(X) = \frac{1}{\lambda}$. The exponential distribution is the only continuous distribution which is memoryless, that is

$$P(X > t + \tau | X > \tau) = P(X > t)$$

If $X_1, \dots, X_n$ are independent exponential random variables with parameters $\lambda_1, \dots, \lambda_n$, respectively, then $\min(X_1, \dots, X_n)$ is also exponentially distributed with parameter $\sum_{i=1}^n \lambda_i$. Furthermore, the probability of X_i taking the minimum is $\frac{\lambda_i}{\sum_{i=1}^n \lambda_i}$ [HP92].

Phase-type distribution

Consider a (CTMC) with n transient states and an absorbing state. The phase-type distribution is the time to absorption in this CTMC. It is determined by a vector of initial probabilities $\vec{\alpha} \in \mathbb{R}^n$ and a matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ describing the transition rates between the transient states.

The probability density function of a random variable $X \sim PH(\vec{\alpha}, \mathbf{T})$ is

$$f(t) = \vec{\alpha} e^{\mathbf{T}t} (-\mathbf{T}) \mathbf{1} \quad t \geq 0$$

where $\mathbf{1} = [1, \dots, 1]^\top$ and $e^{\mathbf{T}t} = \sum_{k=0}^{\infty} \frac{\mathbf{T}^k t^k}{k!}$ is a matrix exponential.

The probability distribution of X is defined as

$$F(t) = 1 - \vec{\alpha} e^{\mathbf{T}t} \mathbf{1} \quad t \geq 0$$

and the k th moment is calculated as

$$E(T^k) = (-1)^k k! \vec{\alpha} \mathbf{T}^{-k} \mathbf{1}$$

There are two special cases of the PH distribution that are particularly important in later sections. The first is when $n = 1$. In this case, $\vec{\alpha} = 1$ and $\mathbf{T} = -\lambda$ and we get back the exponential distribution with parameter λ . The second is when $\mathbf{T}$ is diagonal. In this case, the resulting distribution is a mixture of exponential distributions, and it is called a *hyperexponential distribution*.

Theorem 2.3. *The family of phase-type distributions is dense in the set of non-negative distributions [BN17].*

Matrix-exponential distribution

Due to the way a PH distribution is defined, there are certain constraints on its parameters. Namely, the off-diagonal elements of $\mathbf{T}$ correspond to transition rates in a CMTC, hence, must be non-negative. The matrix-exponential distribution relaxes this condition, generalising the PH distribution.

The notation is $X \sim ME(\vec{\alpha}, \mathbf{T})$ and the formulas for the probability density function $f(t)$, the probability distribution function $F(t)$, and the moments $E_k(X)$ stay identical to that of the PH distribution. Furthermore, as the family of ME distributions contains the family of PH distributions, it is also dense in the family of non-negative distributions.

The difficulty of working with ME distributions comes from the fact that without the non-negativity constraints on $\mathbf{T}$, for a given set of parameters, the resulting probability distribution can be invalid, e.g., it can produce a density function that is not non-negative. Checking the validity of an ME distribution with given parameters is computationally expensive due to the matrix exponential element in the density and distribution functions.

KPC-PH distribution

The KPC-PH distribution is a special class of PH distributions. It is constructed using Kronecker product composition (KPC).

Let $\mathbf{A} \in \mathbb{R}^{p \times q}$ and $\mathbf{B} \in \mathbb{R}^{r \times s}$, the Kronecker product of these two matrices has dimension $pr \times qs$ and it is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1q}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{p1}\mathbf{B} & \dots & a_{pq}\mathbf{B} \end{bmatrix}$$

Note, that in the special case when $p = r = 1$, the above definition yields the Kronecker product of two row vectors, which is itself a row vector.

Let $X_N \sim PH(\vec{\alpha}_N, \mathbf{N})$ be a general PH distribution and $X_H \sim PH(\vec{\alpha}_H, \mathbf{H})$ be a hyperexponential distribution. X_S has a KPC-PH distribution with parameters $\vec{\alpha}_S$ and $\mathbf{S}$ if

$$\begin{aligned}\vec{\alpha}_S &= \vec{\alpha}_N \otimes \vec{\alpha}_H \\ \mathbf{S} &= -\mathbf{N} \otimes \mathbf{H}\end{aligned}\tag{2.1}$$

The fact that X_H corresponds to a hyperexponential distribution ensures that X_S , as defined above, produces a valid PH distribution. KPC-PH distributions composed of $n \geq 2$ components can be recursively defined by repeatedly using the above definition. The composite structure of KPC-PH distributions means that certain useful measures, particularly the moments of the distribution, can be calculated efficiently even when the distribution has many phases. It can be shown that the following relation holds when computing the k th moment of a KPC-PH distribution [CZS10b]

$$E(X_S^k) = \frac{1}{k!} E(X_N^k) E(X_H^k)\tag{2.2}$$

2.4 M/M/1 queues alone and in networks

A queue with i.i.d. exponential inter-arrival times and i.i.d. exponential service times is called an $M/M/1$ queue. The associated stochastic process X_t that takes the value of the length of the queue at time t is a birth-death process, that is, a CTMC in which a one-step transition can only change the current state by one. In this case either a task arrives at the queue or there is a departure when a task has completed its service.

An arrival process with i.i.d. exponential inter-arrival times is also called a Poisson process. Due to the memoryless property of the exponential distribution, the forward recurrence time in a Poisson process — the time until the next arrival — is also exponential and it is independent of the past. An important property of Poisson processes is the *Random Observer Property (ROP)* or *PASTA* (Poisson arrivals see time averages) and it states the following.

Theorem 2.4 (Random Observer Property). *Let $\{X_t | t \geq 0\}$ be a Poisson process. If there is an event at τ , the state changes from $X_{\tau-}$ to $X_{\tau+}$. The random observer property states that the distribution of $X_{\tau-}$ is independent of the event at τ .*

For example $X_{\tau-}$ can be the length of a queue before and $X_{\tau+}$ after the arrival of a new task. The steady-state probability distribution of the queue length in an $M/M/1$ queue with arrival rate λ and service rate μ given $\lambda < \mu$ is

$$\pi_n = (1 - \rho)\rho^n$$

where $\rho = \frac{\lambda}{\mu}$. In this case, the queue length random variable has a geometric distribution with parameter ρ . Note that $\lambda < \mu$ ensures $\vec{\pi}$ is a probability vector, therefore it is a necessary and sufficient condition for the queue to be positive-recurrent.

2.4.1 Generating functions method

An important definition in performance modelling of queues, and more specifically in calculating response times in queues, is the notion of the *tagged job*. Response time is usually calculated from the perspective of an arriving job, that is the tagged job. The timer starts at the instant the tagged job arrives at the queue and stops when the tagged job leaves the queue after being served. The elapsed time is what we call response time. When calculating response time distributions, we often condition on the queue length at the time of the arrival of the tagged job. Now, consider the generating function $G(x)$ of a sequence $(a_n(t))_{n \in \mathbb{N}}$, that is

$$G(x, t) := \sum_{n=0}^{\infty} a_n(t) x^n$$

Suppose $a_n(t)$ above denotes the response time distribution conditioned on the queue length n at the time of arrival, and we are able to obtain the corresponding generating function. Assuming

1. the queue is in a steady state,
2. the Random Observer Property stated in Theorem 2.4 holds,

we can use the steady-state queue-length probabilities for deconditioning the conditional response time distributions. This is because the random observer property ensures that the probability of the queue length being n from the perspective of the arriving job is the same as the steady state probability of a queue of length n from the perspective of a random observer. In fact, when the queue length distribution is geometric, deconditioning w.r.t to the queue length is as easy as evaluating the generating function at $x = \rho$ and multiplying it by $1 - \rho$, i.e., $F(t) = (1 - \rho)G(\rho, t)$.

2.4.2 A single M/M/1-PS queue

In Chapter 3, we first consider a single $M/M/1$ queue with PS discipline. PS is not a new concept; it first appeared in [Kle67]. It was initially thought of as an idealised version of Round Robin (RR) scheduling. It was this close connection to RR that helped it gain popularity originally. The main strengths of RR scheduling are fairness and simplicity. Most notably, it is a simple approximation of CPU scheduling done by operating systems. It is a preemptive discipline where each job gets a fixed amount of server time (i.e., a quantum). If the job is not completed by the end of the quantum, it is interrupted and moved to the end of the queue. Now, assuming that switching from one job to another is instantaneous, if we keep decreasing the quantum, we are approaching processor sharing. Hence, processor sharing can be thought of as the limiting schedule of RR when the quantum approaches zero. Consequently, in a PS queue, the service capacity is equally divided amongst all the customers in the queue. The rate at which a job is being served decreases linearly as the queue length increases. Letting μ be the service rate of the server and n be the number of jobs at the station, each job receives an instantaneous service rate of $\frac{\mu}{n}$. The fairness of PS comes from the fact that the service time spent by a job in the system is proportional to its size.

Since the early days of processor sharing, there turned out to be other areas in computing science where this particular scheduling was applicable and useful for modelling. It became ubiquitous once again with the emergence of web servers, as it was well-suited for modelling the flow-level performance of bandwidth-sharing protocols in packet-switched networks [NMM98, Vis03].

Most recently, it has gained traction due to its widespread usage in modelling cloud computing [ACC⁺14, ACCP12, XWM⁺11] and more specifically in modelling Virtualized Network Functions in cloud computing [GRS19].

As mentioned in the introduction, the PS discipline is significantly more challenging than other disciplines when it comes to response times. The main reason for this is the concept of overtaking. Overtaking occurs when a later arrival can affect the response time of a job that is already in the queue. In the case of PS, this happens because later arrivals affect the queue length, which in turn slows down the service experienced by a particular job. This dependency on future arrivals, along with the fact that the order of customers leaving the queue is not necessarily the same as the order in which they arrive, is what makes it especially difficult to track the system analytically.

2.4.3 Network of M/M/1–PS queues

A pair M/M/1-PS queues in tandem

In Chapter 3, we also consider a pair of $M/M/1$ queues under PS scheduling in tandem. The service time is exponentially distributed with rates μ_1 and μ_2 at queues 1 and 2, respectively. Tasks arrive at the first queue according to a Poisson process with rate λ . After service completion at the first queue, the tasks proceed to the second queue and finally exit the system once they have been served at the second queue.

A pair of JSQ–PS queues

In Chapter 4, the queueing model considered is a small network of queues. Specifically, we consider two parallel $M/M/1$ queues, where each queue has an independent server with exponential service times, and parameters μ_1, μ_2 . The queueing discipline at the individual queues is PS.

The arrivals to the network are i.i.d. and exponentially distributed with rate λ . An arriving job joins the shorter of the two queues; furthermore, if the queues are of equal length, the job

joins queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. In other words, the queueing network consists of a pair of parallel $M/M/1$ queues with JSQ-PS scheduling.

2.5 A single PH/PH/1 queue

A generalisation of the $M/M/1$ queue is achieved by relaxing the assumption of exponential inter-arrival and service times and instead using the more general family of PH distributions. The resulting queue is called a $PH/PH/1$ queue. In this case, the stochastic process tracking the queue length is no longer a CTMC, as the PH distribution is not memoryless, therefore violating the Markov property. However, by expanding the state space to include both the queue length and the phases of the inter-arrival and service times, the resulting stochastic process becomes a CTMC. In this context, the queue length is often referred to as the level of the process. A CTMC that allows for internal phase transitions within a level and in which a process can only move up or down one level is called a *quasi-birth-death (QBD) process*. QBD processes are a much more powerful tool for accurately modelling real-world systems, where the arrival and service time distributions are seldom strictly exponential.

An arrival process that has i.i.d. inter-arrival times that follow a phase-type distribution is a *phase-type renewal process* which represents a special case of Markovian Arrival Processes (MAPs). The PH distribution is typically parameterised by a vector-matrix pair, e.g., $PH(\vec{\alpha}, T)$. However, when considering a phase-type renewal process, it is more convenient to use MAP notation, which simplifies the expression of important properties, such as the generator matrix of the MAP, and consequently, the QBD process. In this new representation, the phase-type renewal process corresponding to the random variable $PH(\vec{\alpha}, T)$ is expressed as $\mathbf{A} = (\mathbf{A}_0, \mathbf{A}_1)$, where $\mathbf{A}_0 = T$ and $\mathbf{A}_1 = -T\mathbf{1}\vec{\alpha}$. In Chapter 5, we utilise queues with KPC-PH service process. Let $\mathbf{N} = (\mathbf{N}_1, \mathbf{N}_2)$ denote a general PH process and $\mathbf{H} = (\mathbf{H}_0, \mathbf{H}_1)$ a hyperexponential process. A KPC-PH process is defined as $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_1)$, where

$$\mathbf{S}_0 = -\mathbf{N}_0 \otimes \mathbf{H}_0$$

$$\mathbf{S}_1 = \mathbf{N}_1 \otimes \mathbf{H}_1$$

We use the notation $S = N \otimes H$ to denote the KPC-PH process S defined above.

Consider a PH renewal process parameterised by (A_0, A_1) . The infinitesimal generator of the corresponding PH arrival process is calculated as

$$Q_A = A_0 + A_1$$

Analogously, for a PH service process, the transition rate matrix is calculated as $Q_S = S_0 + S_1$. Recall that S_0 records the transition rates between the service phases and $S_1 = -S_0 \mathbf{1} \bar{\alpha}_S$, where $\bar{\alpha}_S$ is the vector of initial probabilities of starting in each service phase. It is important to note that vector $\bar{\alpha}_S$ has a special property. Consider an embedded Markov chain where the phases are recorded only at departure instances. The steady-state probability vector of this discrete-time Markov chain is the same vector $\bar{\alpha}_S$. That is, $\bar{\alpha}_S$ is the equilibrium probability vector of the embedded Markov chain at departure instances.

We are now ready to define the transition rate matrix of the QBD process. Let matrix B denote the service rates (backward transitions), matrix L the phase transitions within a level (local transitions) and matrix F the arrival rates (forward transitions). We focus on QBD processes with the following infinitesimal generator

$$Q := \begin{bmatrix} L_0 & F & 0 & \dots \\ B & L & F & \dots \\ 0 & B & L & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad \begin{aligned} B &:= I_a \otimes S_1 \\ L &:= A_0 \oplus S_0 = A_0 \otimes I_s + I_a \otimes S_0 \\ F &:= A_1 \otimes I_s \end{aligned} \quad (2.3)$$

where I_a, I_s are identity matrices of dimension $a \times a$ and $s \times s$, respectively. Matrix $L_0 \in \mathbb{R}^{as \times as}$ is different to L as only the arrival process has internal phase transitions. Hence, L_0 is given by the following formula.

$$L_0 := A_0 \otimes I_s$$

The state space of the QBD process can be described with triplets (A, S, L) , where $A \in \{1, \dots, a\}$ and $S \in \{1, \dots, s\}$ denote the internal phase of the arrival and service processes, respectively, and $L \in \mathbb{N}_0$ denotes the level of the process, which is also the number of customers in the queue.

The level of the QBD process changes only when an arrival (external forward transition) or a departure (external backward transition) occurs.

Analogously to a simple $M/M/1$ queue, as long as the QBD process is positive recurrent, it has a steady state solution [LR99]. The necessary and sufficient condition for a QBD process to be positive recurrent is ensuring $\lambda < \mu$, where λ is the arrival rate (reciprocal of the mean arrival time) and μ is the service rate (reciprocal of the mean service time). This condition can be easily checked upfront. If the stationary distribution exists, it is the only solution to the equation

$$\vec{\pi}_l = \vec{\pi}_0 \mathbf{R}^l \quad (2.4)$$

where $\vec{\pi}_l = (\pi_{l1}, \pi_{l2}, \dots, \pi_{lm})$ denotes the steady state probabilities corresponding to level l and $m = as$ is the number of internal phases within a given level. Furthermore, $\mathbf{R} \in \mathbb{R}^{as \times as}$ is defined as

$$\mathbf{R} = -\mathbf{F}(\mathbf{L} + \mathbf{F}\mathbf{G})^{-1} \quad (2.5)$$

where $\mathbf{G}$ is the minimal non-negative solution to the equation

$$\mathbf{B} + \mathbf{L}\mathbf{G} + \mathbf{F}\mathbf{G}^2 = 0 \quad (2.6)$$

Matrix $\mathbf{G}$ above has a probabilistic interpretation and plays a central role in our proposed fitting algorithm in Chapter 5. Given the state space defined above, the elements of $\mathbf{G}$ correspond to the following downward first-passage times. Let $X_t \in (A, S, L), t \geq 0$ denote a QBD process and let $i = (i_0, i_1, l)$ and $j = (j_0, j_1, l-1)$. Denoting the elements of $\mathbf{G}$ as G_{ij} , the elements of $\mathbf{G}$ correspond to the following conditional probabilities

$$G_{ij} = P(T_{l-1} < \infty \wedge X_{T_{l-1}} = j | X_0 = i) \quad (2.7)$$

where T_{l-1} is the first passage time to level $(l-1)$. Assuming the QBD process is positive recurrent, the process reaches any level in a finite time with probability 1, regardless of the starting state. Therefore the rows of $\mathbf{G}$ correspond to the conditional probabilities that, given the process starts in state i of level l , the first passage to level $l-1$ hits state j of level $l-1$. Since the columns of $\mathbf{G}$ correspond to all possible states of level $l-1$, they sum to 1, making

G a stochastic matrix.

In order to calculate the probability vectors $\vec{\pi}_l$ from (2.4), we also need a formula for $\vec{\pi}_0$. Vector $\vec{\pi}_0$ is the unique solution to

$$\vec{\pi}_0(\mathbf{L}_0 + \mathbf{F}\mathbf{G}) = 0 \quad (2.8)$$

$$\vec{\pi}_0(\mathbf{I} - \mathbf{R})^{-1}\mathbf{1} = 1 \quad (2.9)$$

where (2.9) is an equivalent condition to $\vec{\pi}\mathbf{1} = 1$.

The $PH/PH/1$ queues used in Chapter 5 employ the First come first served (FCFS) discipline, where customers are serviced non-preemptively in the order of arrival to the queue.

In Chapter 5, we also use ME distributions to fit real-life traces, resulting in $PH/ME/1$ queues. The use of ME distributions as the service distribution is feasible because queueing results stated for PH distributions stay intact when using ME distributions [BN12].

2.6 Complex analysis

In this section, we include a list of definitions and theorems from complex analysis that are used throughout the thesis. We start by defining two complex transforms corresponding to random variables.

Let $F : \mathbb{R}_0^+ \rightarrow \mathbb{R}$, then the *Laplace-Stieltjes transform* of F is a function itself $\mathcal{LS}\{F\} : \mathbb{C} \rightarrow \mathbb{C}$

$$\mathcal{LS}\{F\}(s) := \int_0^\infty e^{-st} dF(t)$$

Whereas, given a function $f : \mathbb{R}_0^+ \rightarrow \mathbb{R}$ the *Laplace transform* of f , $\mathcal{L}\{f\} : \mathbb{C} \rightarrow \mathbb{C}$ is defined as

$$\mathcal{L}\{f\}(s) = \int_0^\infty e^{-st} f(t) dt$$

Hence, if X is a random variable with distribution function F and density f , then the Laplace-Stieltjes transform of F and the Laplace transform of f coincide, that is

$$\mathcal{LS}\{F\}(s) = \mathcal{L}\{f\}(s) = E(e^{-sX})$$

When calculating the response time distribution in a queue it is very common to obtain the Laplace-Stieltjes transform of the distribution, instead of the density itself, which then needs to be inverted. The following theorem ensures that the Laplace-Stieltjes transform defines the distribution uniquely almost everywhere.

Theorem 2.5 (Inversion of the Laplace-Stieltjes transform). *If there are two functions with the same Laplace-Stieltjes transform, they can only differ on a set which has Lebesgue measure zero. Hence, the inverse function is uniquely determined Lebesgue almost everywhere.*

General properties of the Laplace-Stieltjes transform:

1. **linearity:** $\mathcal{LS}\{aF_1 + bF_2\} = a\mathcal{LS}\{F_1\} + b\mathcal{LS}\{F_2\}$
2. **derivative:** if F has derivative F' , the Laplace-Stieltjes transform of F' is

$$\mathcal{LS}\{F'\}(s) = s\mathcal{LS}\{F\}(s) - F(0)$$

3. **integral:** the Laplace-Stieltjes transform of $\int_0^t F(\tau)d\tau$ is

$$\mathcal{LS}\left\{\int_0^t F(\tau)d\tau\right\} = \frac{1}{s}\mathcal{LS}\{F\}(s)$$

4. **convolution:** the Laplace transform of $(f * g)(t) = \int_0^t f(\tau)g(\tau - t)d\tau$ is

$$\mathcal{LS}\{(f * g)(t)\} = \mathcal{LS}\{F\}\mathcal{LS}\{G\}$$

One of the main reasons one might decide to calculate the Laplace-Stieltjes transform of a distribution is due to the derivative property above. Whenever derivatives arise in a functional equation, it is very convenient to take advantage of the linear connection between the LST of the original function and its derivative.

The second transform we utilise is the *Mellin transform*. If X is a non-negative random variable, its Mellin transform $\mathcal{M}_X(s) : \mathbb{C} \rightarrow \mathbb{C}$ is defined as follows

$$\mathcal{M}_X(s) = E(X^{s-1})$$

The Mellin-transform evaluated at positive whole numbers returns the moments of the random variable. Furthermore, if X and Y are two independent random variables and $Z = XY$, we have the following factorisation property

$$\mathcal{M}_Z(s) = \mathcal{M}_X(s)\mathcal{M}_Y(s)$$

Next, we define the complex logarithm and exponential functions and state the conditions under which these functions are holomorphic.

Let $w \in \mathbb{C}$, the *complex logarithm* of w is defined as

$$\log w = \log_{\mathbb{R}} |w| + i \arg w$$

where $\log_{\mathbb{R}}$ denotes the known definition of logarithm for real numbers. Note that by definition the complex logarithm is a multi-valued function (since $\arg w$ is multi-valued).

Theorem 2.6. *If D is a simply connected set and $0 \notin D$ then there is a holomorphic branch of $\log z$ on D .*

Theorem 2.7. *If D is a simply connected set and $f : D \rightarrow \mathbb{C}$ holomorphic and $f \neq 0$ then there is a holomorphic branch of $\log f$ on D .*

The theorems above state that even though the logarithm is multi-valued, if the domain is restricted in a way that going around 0 is not possible, the logarithm can be defined consistently on the domain as a holomorphic function, in particular, it is single-valued.

Let $z, w \in \mathbb{C}$, $z \neq 0$ the *complex exponential* function is defined as

$$z^w = e^{w \log z}$$

Therefore, whenever a holomorphic branch of the complex logarithm is defined on a domain, it follows that the complex exponential function is also single-valued and holomorphic on the same domain.

Finally, we state two integral theorems that show how a function at any point inside a disk

can be calculated from its values on the boundary of the disk, and a theorem of continuous functions on compact sets.

Theorem 2.8 (Cauchy's Integral Formula). *Let $U \subseteq \mathbb{C}$ be an open set and $f : \mathbb{C} \rightarrow \mathbb{C}$ be holomorphic on U and let the closed disk $D := \{z \mid |z - c| \leq r\}$ be a subset of U . Then for all $z_0 \in \text{Int}D$,*

$$\frac{1}{2\pi i} f(z_0) = \int_{|z-c|=r} \frac{f(z)}{z - z_0} dz$$

Theorem 2.9 (Cauchy's Differentiation Theorem). *Let $U \subseteq \mathbb{C}$ be an open set and $f : \mathbb{C} \rightarrow \mathbb{C}$ be holomorphic on U and let the closed disk $D := \{z \mid |z - c| \leq r\}$ be a subset of U . Then for all $z_0 \in \text{Int}D$,*

$$f^{(n)}(z_0) = \frac{n!}{2\pi i} \int_{|z-c|=r} \frac{f(z)}{(z - z_0)^{n+1}} dz \quad 0 \leq n$$

Theorem 2.10 (Heine-Cantor Theorem). *Let $f(z) : D \rightarrow \mathbb{C}$ where $D \subset \mathbb{C}$ is a compact set. If f is continuous on D then f is also uniformly continuous on D .*

2.7 Related work

This section provides an overview of the most important results related to response time distributions and fitting algorithms. We begin by reviewing the main results on response time distributions for both single queues and queueing networks, with a particular focus on response times in PS queues, along with the methods used for obtaining these results. We then discuss results concerning the optimality of PS and JSQ scheduling. Finally, we present an overview of the different approaches used to fit distributions to traces, outline the advantages and disadvantages of each approach, and explain how the fitting algorithms proposed in Chapter 5 differ from existing methods.

2.7.1 Response time distributions

Response time distributions are usually derived by obtaining a functional equation for the LST of the response time distribution, most frequently by applying the generating functions method

described in Section 2.4 at some stage of the process. We demonstrate how existing approaches are closely linked to the problem considered and, therefore, typically leverage specific properties of the underlying system to derive the solution, providing little scope for reusing these methods to address new queueing problems. This is in contrast with the numerical approximation technique used in this dissertation, which is first used to solve the $M/M/1$ -PS queue — a problem with a known analytical solution that allows for detailed evaluation of the method — and then adopted to compute the response time in a tandem pair of $M/M/1$ -PS queues and a pair of JSQ-PS queues, both of which are solved here for the first time.

Single queue models

In the case of a single queue, obtaining the functional equations themselves is usually straightforward; however, finding a solution to the equations requires a case-specific approach, even in this case. In [CJMT70], the LST was calculated for the response time distribution conditioned on the service time of the tagged job for an $M/M/1$ queue under the PS discipline. The functional equation obtained for the LST is

$$\frac{\partial W}{\partial \tau}(s, \tau, z) - (\mu z^2 - (\lambda + \mu + s)z + \lambda) \frac{\partial W}{\partial z}(s, \tau, z) = (\mu z - \lambda)W(s, \tau, z)$$

The authors convert the above partial differential equation into a system of ordinary differential equations. These are solved directly using the integrating factor method. However, the obtained formula contains an arbitrary function f . The key to the solution is finding the boundary value condition $W(s, 0, z) = \frac{1}{1-z}$, which allows f to be determined. The formula obtained for the LST of conditional response time is

$$W^*(s|\tau) = \frac{(1-\rho)e^{-(\lambda(1-r)+s)\tau}(1-\rho r^2)}{-\rho(1-r)^2 e^{-\frac{\mu\tau}{r} + \lambda r\tau} + (r\rho - 1)^2} \quad (2.10)$$

where $r = \frac{(\lambda+\mu+s) - \sqrt{(\lambda+\mu+s)^2 - 4\lambda\mu}}{2\lambda}$. However, the above formula is derived under the assumption that s is a positive real number, making its inversion non-trivial. The inversion of the LST is not attempted by the authors.

Approximately 15 years later, this issue is resolved in [Mor85]. Deconditioning the above formula w.r.t. the tagged task service time, assuming this is exponential with parameter μ , the LST of the unconditional response time distribution is

$$W^*(s) = \int_0^\infty W^*(s|\tau) \mu e^{-\mu\tau} d\tau = \int_0^\infty \frac{(\mu - \lambda) e^{-(\lambda(1-r)+s+\mu)\tau} (1 - \rho r^2)}{-\rho(1-r)^2 e^{-\frac{\mu\tau}{r} + \lambda r\tau} + (r\rho - 1)^2} d\tau \quad (2.11)$$

[Mor85] is able to extend this formula to the complex s -plane with a carefully chosen branch cut to eliminate the poles. The curve along which the inversion is taken is transformed into a curve around the cut and the following integral is obtained

$$P(W > t) = 2 \int_0^\pi \frac{\exp \{ -\theta [2\sqrt{\rho} - (1 + \rho) \cos \theta] / (1 - \rho) \sin \theta - (1 - \rho)^2 t / (1 + \rho - 2\sqrt{\rho} \cos \theta) \}}{(1 - \rho)(1 + \exp \{ -\pi [2\sqrt{\rho} - (1 + \rho) \cos \theta] / (1 - \rho) \sin \theta \})} \sin \theta d\theta$$

The result is laborious and involves several intricate analytic transformations. It is worth noting that once complex inputs are considered for the LST — enabling efficient inversion of the transform — maintaining the rigour of the proofs requires significantly more effort and care. [Mor85] also examines asymptotic behaviours in the case of heavy traffic.

[HP93] calculates the LST of response time distribution in an $M/M/1$ queue that allows for negative customers. In the case of FCFS and LCFS discipline, the resulting functional equations are solved using an analyticity argument. Specifically, the solution is found by exploiting the analyticity of the generating function of conditional response time LSTs. To illustrate the method, we consider the functional equation obtained in the FCFS case

$$R(x, y)G(x, y, s) = \frac{\mu}{1 - y} - \frac{\lambda^+}{y}G(x, 0, s)$$

where $R(x, y) = s + \lambda + \mu(1 - x) - \frac{\lambda^+}{y} - \lambda^- y$. As $G(x, y, s)$ is a generating function, it must be analytic everywhere inside a unit circle, therefore whenever $R(x, y)$ has a root, the right-hand side of the equation must also vanish. Using Rouché's theorem, the authors find a root of $R(x, y)$ inside the unit circle and are therefore able to find a solution for $G(x, 0, s)$, which

is enough to obtain the LST of response time in this particular queueing model, as it can be calculated as $W^*(s) = (1 - \rho)G(\rho, 0, s)$.

The functional equation is more complicated when considering the PS discipline but it can be handled analogously to [CJMT70]. That is, the partial differential equation

$$(\mu^- z^2 - (s + \lambda + \mu)z + \lambda^+) \frac{\partial}{\partial z} G - \frac{\partial}{\partial x} G = (s + \lambda - \mu^- z)G$$

is solved by obtaining the solution to the auxiliary equations, a unique solution again requiring the boundary condition $G(z, s, 0) = 1/(1 - z)$ as before. The obtained LST is

$$W^*(s) = \frac{\mu}{\lambda^- + \mu} \frac{(1 - \rho)(1 - \rho r^2)r}{(1 - \rho r)^2} \sum_{n=0}^{\infty} \left(\frac{\rho(1 - r)^2}{(1 - \rho r)^2} \right)^n \frac{1}{1 + n(1 - \rho r^2)}$$

where r is the smaller root of the quadratic equation $\lambda^+ z^2 - (s + \lambda + \mu)z + (\mu + \lambda^-) = 0$. The inversion of the LST is again non-trivial due to the infinite sum component and it is not attempted in the paper. Instead, the moments of the response time are calculated.

One key takeaway from the above examples is that studying a queue under the PS discipline introduces derivatives or partial derivatives in the functional equation of the generating function. This, in turn, complicates the process of solving the resulting functional equations and the obtained solution may not be straightforwardly invertible, which leaves the problem of deriving the response time distribution unresolved.

In [ZK09b], the original $M/M/1$ queue with processor sharing is revisited. However, instead of obtaining the LST conditioned on the service time of the tagged job as in [CJMT70], the focus shifts to the LST conditioned on the number of tasks in the system at the time of the arrival of the tagged job. The approach is again case-specific and it uses discrete Green's functions to obtain the LST. The authors proceed with studying limit behaviours, such as the queue length tending to infinity or the heavy traffic behaviour. Later, the results are extended to an $M/M/1$ queue under PS discipline with finite capacity [ZK09a]. A relatively recent development is found in [GRS18], where a formula for the response time distribution in a PS queue with batch arrivals is derived under the assumption that the batch size distribution is

geometric. Using analyticity properties of the generating function of conditional response time distributions, the authors are able to determine a boundary term for this generating function. Using the boundary term, the unique solution to the partial differential equation governing the generating function is derived in terms of its one-sided LST. This is similar to the method used in [HP93]. However, the detailed steps required to obtain a solution are more complex and rely on the analytic properties of certain functions within the equation.

Next, we present results that are limited to the moments of the response time distributions. These are included for completeness but without a detailed discussion of the methods used. A technique for asymptotic expansions of the first two moments in both open and closed queueing networks with multiple job classes, under normal traffic, was developed in [MM83], and later extended to heavy traffic in [MM85]. Subsequently, [LW13] develops an algebraic method to calculate the moments of the conditional response time in an $M/M/1$ queue (again conditioning on the queue length at the time of the arrival of the tagged job). The method applies to multiple job classes.

Finally, two examples are presented where response time distributions are studied via asymptotic methods instead of exact results. These are [Kne93], which proposes an asymptotic solution to estimate the response time distributions in an $M/M/1$ queue with finite capacity, and [Har08], which is particularly interesting as instead of obtaining and inverting the Laplace transform (as typically done), a uniform estimation of the response time in an $M/G/1$ queue is achieved through the decomposition of the transition state into two separate components: a service component and an arrival-departure component. Under very light or very heavy load, the contribution of the second component vanishes, and the first component is used to estimate the true response time. An upper bound on the moments of the original system is also provided.

As one of the main goals of this dissertation is to study the PS discipline, it is worth discussing the optimality of this scheduling algorithm, in terms of the response time distribution, within the context of single-queue systems. In [Sch68, Smi78], it is shown that, for any arrival process and service time distribution, the scheduling policy that minimises the mean response time is the shortest remaining processing time (SRPT) policy. However, as noted by [BHB01], despite

this well-known result, SRPT is rarely used in practice, since its improvements in mean response time come at the cost of penalising large jobs in favour of shorter ones. Consequently, a commonly used alternative is the PS discipline, which is generally considered fairer in the sense that the expected mean slowdown — defined as the average ratio of the response time of a job to its service requirement — is the same for every job, regardless of its size. [BHB01] formally analyses the unfairness of SRPT and concludes that, under certain conditions (e.g., heavy-tailed job size distributions, or light load conditions), this unfairness vanishes. Furthermore, it shows that SRPT achieves a lower mean slowdown than PS. Another reason why PS remains ubiquitous, despite its suboptimality, is that it does not require tracking the remaining processing time of each job in the queue, thereby reducing overhead, nor does it require prior knowledge of job sizes at arrival, which is unavailable in some systems.

Queueing networks

In the case of a network of queues, even obtaining the functional equations can impose significant challenges. A method that has been repeatedly used to successfully obtain the LST of response time distribution in a network of queues is the reversibility argument. It first appeared in [BKK84], where the LST of response time was obtained for a closed cyclic queueing network of M exponential FCFS queues. The main idea is as follows. If a CTMC is reversible, the reversed process has the same stationary distribution as the original process, albeit with different transition rates. Conditioning on the state at the time the tagged task leaves the first queue, the response time at the reversed first queue is independent of the remaining sojourn time at queues 2 to M . Therefore, the two sojourn times can be calculated separately and taking the product of the LSTs we get the LST of the total sojourn time.

The most basic open queueing network consists of two queues arranged in tandem. The model is described at the end of Section 2.4.2. The response time distribution in a tandem pair of FCFS queues with batch processing where the batch sizes are finite, and the inter-arrival and service times are exponential has been obtained in [HB21b]. The authors use the reversibility argument and condition on the state at the time the batch of the tagged task leaves the first queue. The LST of response time distribution at the second queue is then derived using a

similar analyticity argument as seen in [HP93], using Rouché's theorem to find the roots of certain functions, albeit the problem is more complex as it requires the solution of a vector of generating functions. Furthermore, due to the batch departures, the sojourn time at the second queue depends on the length of the first queue despite the FCFS scheduling, making the deconditioning of conditional response times non-trivial. The LST of the unconditional response time distribution is successfully obtained via the evaluation of a certain complex integral.

The reversibility argument is used once again, to obtain the LST of response time distribution in a tandem network of G-queues, i.e. queues that allow negative arrivals, in [HP95]. The problem is reduced to a boundary value problem (BVP), which in turn reduces to a Fredholm integral equation of the second kind that can be solved numerically. Although the Laplace Stieltjes transform or its inversion has not been computed numerically in the paper, it can be evaluated using standard numerical software.

Another example, where the solution to the functional equations of the response time LST is reduced to a BVP is [CJFM86], where functional equations for the LST of the response time distribution are obtained for a PS and FCFS queue in tandem, as well as for two PS queues in tandem. In this case, the BVPs reduce to Fredholm integral equations of the first kind. These are commonly more difficult to solve than those of the second kind. Consequently, the numerical inversion to obtain explicit solutions for the functional equations of the LST of response time distributions was unsuccessful, and the problems remained unsolved. This example illustrates that even when functional equations are successfully obtained in a network where some queues are under the PS discipline, the solution of these functional equations can be very challenging. In the case of a PS-FCFS tandem, the first two moments are approximated in [CC90], and more recently, [Har24] obtained the LST of the response time distribution for this model using a novel numerical approximation method. The results are compared against simulation and appear to be accurate.

Another important open queueing network is the JSQ network, described in Section 2.4.2 in the case of two queues under the PS discipline. JSQ scheduling was first considered almost 70 years ago in the symmetric case, when the queueing discipline is FCFS [Kin61], i.e. where $a_1 = a_2 = 1/2$ and $\mu_1 = \mu_2$. A complex closed-form solution for the queue-length distribu-

tion and mean response time, in this case, is obtained in [FM77], using analyticity arguments. Specifically, similar to the FCFS and LCFS cases in [HP93], the numerator of a certain expression must vanish whenever the denominator does, as an analytic function, by definition, cannot have poles within its domain of analyticity. The analysis requires a lot of effort, and it does not generalise to the case where $\mu_1 \neq \mu_2$. Under the FCFS discipline, once a customer joins the queue, no later-arriving customer can overtake it. This means that in order to calculate the response time distribution, there is no need to track the customers joining the queue after said customer. Therefore, the calculation of response time distribution in a JSQ-FCFS queueing model is straightforwardly reduced to obtaining just the equilibrium queue length probabilities. Functional equations leading to the generating function of the equilibrium queue-length probabilities for a pair of JSQ-FCFS queues have been obtained and solved recently in [Har24], using the same method applied to derive the response time in a PS-FCFS tandem in the same paper. The results again demonstrate good accuracy when compared to simulation. Finally, [GOST15] studies the queue-length distribution of two parallel $M/M/\infty$ queues operating under JSQ scheduling, where ties are broken with equal probability. When the service rates at the two queues are identical, the problem is reduced to solving a Fredholm integral of the first kind, and the solution is expressed in terms of Legendre polynomials.

It is again interesting to consider the optimality of JSQ scheduling, as Chapter 4 studies the response time distribution of two parallel PS queues under JSQ scheduling. In a system of several FCFS queues with Poisson arrivals and identical exponentially distributed servers, [Win77] proves that JSQ scheduling is optimal in the sense that it maximises the number of customers completing service within a given time. [Web78] extended this result by removing the assumptions on the arrival process and showing that the policy remains optimal for any service distribution with a non-decreasing hazard rate. However, when job sizes are highly variable, JSQ is far from being the optimal routing strategy [HBCM99, HB00].

Few works study the optimality of JSQ scheduling of parallel PS queues; however, [Bon02] shows via simulation that, in the case of two parallel queues with exponential service distributions, among scheduling policies that rely solely on the queue-length at the servers, JSQ achieves the lowest mean response time. [GBSW07] provides a more extensive analysis for a system

with several PS queues and general service time distributions, first deriving an approximate solution for the queue-length distribution and then studying the optimality of JSQ scheduling for parallel PS queues in terms of the mean queue-length. They compare JSQ with several other policies, including Least-Work-Left (LWL) and OPT-0, where each arriving job is assigned to minimise the mean response time of all jobs currently in the system, assuming no future arrivals. Across all service distributions considered, OPT-0 is the most optimal policy; however, JSQ performs very closely, within 10% of OPT-0. This is significant, as both LWL and OPT-0 require knowledge of job sizes, whereas JSQ does not.

In Chapter 3, we first derive and solve a new functional equation for the LST of the response time distribution in an $M/M/1$ -PS queue without conditioning on the service time of the tagged job. Using our method, the obtained LST is straightforwardly inverted, in contrast to [CJMT70], where the inversion of the formula was only achieved later in [Mor85]. This type of simplification is important when seeking exact analytic solutions for more complex systems in which the $M/M/1$ -PS queue serves as a building block, such as two $M/M/1$ -PS queues in tandem. Next, we adopt the method used in [Har24] to solve the functional equation numerically in the $M/M/1$ -PS case, providing further validation of the approach by comparing it against exact results. We then extend this approach to derive the response time density in a PS-PS tandem and in a pair of JSQ-PS queues, both for the first time.

2.7.2 Fitting algorithms

Queueing performance is well understood in the case of Markovian representations of inter-arrival and service times. Therefore, most fitting algorithms use phase-type (PH) distributions or Markovian arrival processes (MAPs) to fit trace data. PH distributions and MAPs can be readily plugged into various families of performance models, including quasi-birth death (QBD) processes, queueing networks, stochastic Petri nets, and Markovian process algebras [RIS22, BC17, BGDMT06, RS02, Buc94, RS07]. PH distribution fitting, in particular, has received considerable attention over the years due to its importance in modelling and simulation studies,

software reliability, web server modelling, load balancing, and risk theory [NT13, YHB12, BN17, RH12, ODT13, PSCS⁺11, CRS01, Bla05, APYS22].

A fitting problem can be described as follows. Let x be the set of samples collected from a distribution. A PH fitting algorithm aims to find the set of parameters $(\vec{\alpha}, T)$ such that the corresponding random variable $X \sim PH(\vec{\alpha}, T)$ generates a sample that closely matches x statistically. Established fitting approaches include the expectation-maximization (EM) algorithm, cumulative distribution function (CDF) fitting, and moment matching algorithms [TBT06, FW98, BHT05]. We give details of all three approaches along with their advantages and drawbacks.

EM algorithms

The EM algorithm aims to calculate the maximum likelihood estimate for a stochastic model that depends on latent variables, which are not directly observable. For instance, in a queue with PH-distributed service times, the departure times from the queue can be recorded; however, the specific phases a task passes through before it departs cannot be observed. Let x be the data corresponding to a complete observation of a random variable X with density f_θ , where θ is the set of parameters to be estimated. In the case of a PH distribution, θ records the vector of starting probabilities and the transition rate matrix. The observed random variable is $Y = u(X)$ with density g_θ , where u is a many-to-one mapping and the corresponding dataset is y . The likelihood function $L(x, \theta) = f_\theta(x)$ cannot be maximised directly, as only the data y is available. The EM algorithm aims to maximise the log-likelihood function, by performing the following two steps in each iteration. Step 1 is the E-step, in which the conditional expectation of the log-likelihood

$$E_{\theta_i}(\log f_\theta(X) | u(Y) = y)$$

is calculated using the current estimates θ_i . Step 2 is the M-step, which finds the θ values that maximise the conditional expectation, and these parameters form the new estimate θ_{i+1} . It can be shown that this iteration increases the likelihood $g_\theta(y)$ in each iteration. However, the convergence of θ_i is often hard to prove, furthermore, it is not guaranteed to be to the

maximum likelihood $\hat{\theta}$.

There are several examples of EM-based algorithms for fitting PH distributions. In [ANO96], the density function of a complete observation given a PH distribution with starting probabilities $\vec{\pi}$ and transition rate matrix T is calculated as

$$f(\vec{x}; \vec{\pi}, T) = \prod_{i=1}^p \pi_i^{B_i} \prod_{i=1}^p \exp(t_{ii} Z_i) \prod_{i=1}^p \prod_{\substack{j=Q \\ j \neq i}}^p t_{ij}^{N_{ij}}$$

where B_i is the number of Markov processes starting in state i , Z_i is the total time spent in state i and N_{ij} is the total number of jumps from state i to state j . In the E step, the values of B_i , Z_i and N_{ij} are updated by calculating the corresponding conditional expectations using the current estimates for $\vec{\pi}$ and T . In the M step, the maximum likelihood estimate of the parameters of the PH distributions is calculated as

$$\hat{\pi}_i = \frac{B_i}{n} \quad \hat{t}_{ij} = \frac{N_{ij}}{Z_i} \quad \hat{t}_{ii} = - \left(\frac{N_{i0}}{Z_i} + \sum_{\substack{j=1 \\ j \neq i}}^p \hat{t}_{ij} \right)$$

An EM algorithm is used to fit a mixture of Erlang distributions in [TBT06]. The expectation of the conditional log-likelihood is calculated as

$$Q(\Theta, \hat{\Theta}) = \sum_{m=1}^M \sum_{k=1}^K \log(\alpha_m) \cdot q(m | y_k, \hat{\Theta}) + \sum_{m=1}^M \sum_{k=1}^K \log(p_m(y_k | \theta_m)) \cdot q(m | y_k, \hat{\Theta})$$

where $\hat{\Theta} = (\alpha_1, \dots, \alpha_M, \theta_1, \dots, \theta_M)$ is the current set of parameters, $p_m(y_k | \theta_m)$ is the density of the m th Erlang distribution in the mixture and $q(m | y_k, \hat{\Theta})$ is the probability of the k th observation y_k being generated by the m th Erlang component of the mixture. The authors then derive formulas to calculate the new values for α_m and θ_m , $1 \leq m \leq M$ that maximise $Q(\Theta, \hat{\Theta})$, e.g., the new estimate for α_m is given by $\alpha_m = \frac{1}{K} \sum_{k=1}^K q(m | y_k, \hat{\Theta})$. [BEN11] derives the formula to calculate the Fisher information matrix for the EM algorithm for general PH distributions. EM algorithms are understood to perform well on shorter traces. However, a significant drawback of the approach is that EM algorithms tend to converge slowly, requiring many iterations. Additionally, both steps in the iteration can be computationally expensive, particularly when

fitting PH distributions with many phases. As a result, these algorithms are computationally costly, requiring long-running executions to find accurate PH distributions. Consequently, EM algorithms are often overly demanding for fitting the long traces necessary for heavy-tailed distributions. An improvement on traditional EM-algorithms is [RDS04] which proposes a compositional approach in which the trace is partitioned into smaller chunks. Each partition is fitted individually using the EM algorithm and the resulting distributions are subsequently combined. The proposed algorithm can handle longer traces than traditional EM algorithms; however, it remains computationally expensive.

Matching based algorithms

Instead of working with the whole trace as EM algorithms do, matching-based algorithms use certain measures of the distribution as their input, such as the moments or the CDF function. As a result, compared to EM algorithms, matching-based algorithms are less costly to run. The first step of a matching-based approach is to calculate an estimate for the selected measure from the input trace, e.g., the empirical CDF (ECDF) or empirical moments. An immediate drawback of these methods is that the empirical values used to find a well-fitting distribution might be inaccurate. Higher moments, in particular, are especially sensitive to the quality of the trace data. These algorithms then typically require inverting analytical expressions that describe the chosen statistics of the PH distribution, such as high-order polynomials used to compute the moments of PH distributions.

[BHT05] calculates bounds on the second and third moments of an acyclic PH (APH) distribution and uses these constraints to match the first three moments of a trace to a minimal APH distribution. That is, let n_2 and n_3 be the normalised second and third moments of an APH distribution with n phases, then the lower bound on n_3 is

$$\begin{cases} l_n \leq n_3 & \text{if } \frac{n+1}{n} \leq n_2 \leq \frac{n+4}{n+1} \\ \frac{n+1}{n}n_2 < n_3 & \text{if } \frac{n+4}{n+1} \leq n_2 \end{cases}$$

where l_n is given in [BHT05][Equation 4]. A similar expression is derived for the upper bound. [Hor13] uses generalised hyper-Erlang distributions to fit the moments of a trace. The computational complexity of expressions, such as moments, is directly proportional to the size of the corresponding PH distribution. Therefore, methods that rely on them tend to impose significant restrictions on the size of the PH distribution they attempt to fit. To address this limitation, compositional approaches have been developed that define large distributions in terms of smaller PH components. For example, [CZS10a] employs distributions constructed using Kronecker product composition to fit the moments of a trace, allowing the fitting of PH distributions of larger size. When relying solely on moments to fit a trace, another issue arises: fitting a small number of moments does not necessarily lead to an equally accurate approximation of the entire distribution. This is because a limited set of moments may fail to capture sufficient information about the characteristics of the underlying distribution. An alternative approach is to use the CDF of the trace for the fitting as in [FW98], however, the constraints on the size of the distribution remain. An interesting recent approach uses Bernstein exponentials [HV23] of the form

$$\text{BE}_n f(x) = \sum_{i=0}^n f\left(-\log\left(\frac{i}{n}\right)\right) \cdot \binom{n}{i} e^{-ix} (1 - e^{-x})^{n-i}$$

which form a subclass of APH distributions, to fit a distribution by sampling its density function f . The method allows for tunable accuracy, notably enabling even the fitting of bounded uniform distributions; however, it can converge slowly and require very large PH distributions that may make the following queueing analysis intractable.

We note that all the above methods fit traces directly, and the resulting distributions are then plugged into existing queueing models. However, collecting service-level data from a real-life model is not always possible. Consequently, in Chapter 5, we propose a novel compositional PH-fitting algorithm that uses the arrival trace and the queue-length trace, both of which are more easily observable, to find the parameters for a service distribution indirectly. Furthermore, our method leverages a compositional approach to enable the fitting of large distributions and combines the CDF with the empirical moments of the given trace, thereby mitigating the

problem of fitting based solely on empirical moments, which tend to be inaccurate for higher moments. Additionally, while there are a lot of PH-fitting algorithms available in the literature, there are very few that use ME distributions for fitting. This is due to the difficulty explained in Section 2.3 regarding checking the validity of an ME distribution with a given set of parameters. One example of an ME fitting algorithm using an exact moment matching technique is [HT16], which is restricted to ME distributions with a handful of states. LSTs are used for fitting in [HM98]; however, it is not guaranteed that the resulting LST belongs to a valid distribution. This issue is resolved in [Fac05], but the resulting algorithm converges even slower than EM algorithms. Thus, we develop a novel compositional ME-fitting algorithm that guarantees the validity of the fitted distribution.

Chapter 3

Response time in a PS queue and two PS queues in tandem

3.1 Introduction

Calculating the response time distribution under PS is a challenging problem. Especially, in the case of networks of queues under PS where existing results are very limited, even in the simplest case of two PS queues in tandem. In this chapter, we first revisit the M/M/1-PS queue and take an alternative approach to derive the LST of its response time distribution. Our method computes the LST directly, without conditioning on the service time of the tagged task. This simplification reduces the complexity of the functional equation, effectively eliminating one variable and transforming the partial differential equation (PDE) derived in [CJMT70] into an ordinary differential equation (ODE). We believe that this type of simplification is a crucial step toward solving more complex models with multiple PS queues. The ODE is solved explicitly, yielding an exact result for the LST of the response time distribution in a form that can be easily inverted to obtain the response time density. Numerical results show an exact match to the LST obtained in [CJMT70].

Next, we study the approximation method proposed in [Har24] to obtain the response time in a tandem pair of PS-FCFS queues, and a pair of JSQ-FCFS queues. In both cases, the results are

compared against simulations; however, simulated results are themselves approximations. This becomes evident when looking at the simulated moment for the PS-FCFS queue, which should match the first moment in an FCFS-FCFS network — calculable exactly using Little's law — yet the obtained values do not align perfectly. Therefore, to further validate the approximation method, we apply it to the functional equation derived for the M/M/1-PS queue and compare its results to the exact solution. We demonstrate that a simple transformation of the functional equation enhances the numerical accuracy of the approximation method. Additionally, we derive a recursive formula for efficiently computing response time moments, which proves to be highly accurate.

Finally, we derive functional equations for the PS-PS tandem network using the standard approach, as seen in [CJFM86], while applying the same transformation used in the M/M/1-PS case. Unlike in [CJFM86], where explicit solutions to these equations are not obtained, we successfully apply the approximation method to solve them. This allows us to derive the first solution for the response time density in a PS-PS tandem. Additionally, we derive a recursive formula for efficiently computing response time moments. Both the response time density and the first four moments demonstrate high accuracy when compared to simulation results.

The rest of the chapter is organised as follows. In Section 3.2, the functional equations are derived and solved exactly for an M/M/1-PS queue. In Section 3.3 the functional equations are solved using the approximation method, and the obtained results are evaluated extensively at multiple stages of the approximation. In Section 3.4 the functional equations for a PS-PS tandem are derived and solved using the approximation method. A summary of the main results of the chapter is given in Section 3.5.

3.2 M/M/1-PS: exact solution

In this section, we derive a functional equation for the generating function of the LST of response time distribution in an M/M/1-PS queue and obtain an exact solution for the generating function. Let T be the sojourn time of the tagged job and N denote the queue length at the time of arrival. Furthermore, let $F_n(t) := P(T \leq t | N = n)$ be the conditional distribution function

with corresponding Laplace-Stieltjes transform $L_n^*(s)$. Finally, let $G(z, s) := \sum_{n=0}^{\infty} L_n^*(s)z^n$ be the generating function of the LST of conditional distributions.

Since the queue-length distribution is known to be geometric with parameter ρ , the LST of the response time can be expressed in terms of the generating function as $W(s) = (1 - \rho)G(\rho, s)$.

Proposition 3.2.1. *$G(z, s)$ is governed by the following ordinary differential equation*

$$(\lambda + \mu + s - \mu z)G(z, s) = (\lambda - (\lambda + \mu + s)z + \mu z^2)\frac{\partial G}{\partial z}(z, s) + \frac{\mu}{1 - z} \quad (3.1)$$

where λ is the arrival rate and μ is the service rate of the M/M/1 queue.

Proof. Let the tagged job arrive at the queue at time 0, facing a queue length of N on its arrival, and denote its response time with T . We consider the time interval $(0, t + h)$ and an initial infinitesimal interval $(0, h)$. During this infinitesimal interval, the following events can occur to order h :

1. a new task arrives at the queue with probability λh ;
2. a task other than the tagged is completed with probability $\frac{n}{n+1}\mu h$;
3. the tagged job is completed with probability $\frac{1}{n+1}\mu h$.

Therefore, we have

$$\begin{aligned} P(T \leq t + h | N = n) &= \lambda h P(T \leq t | N = n + 1) + \frac{n}{n+1}\mu h P(T \leq t | N = n - 1) + \\ &\quad \frac{1}{n+1}\mu h 1 + [1 - \lambda h - \mu h] P(T \leq t | N = n) \end{aligned}$$

Taking the limit $h \rightarrow 0$ and after some rearrangement, we get

$$(n+1)\frac{\partial F_n}{\partial t} = -(\lambda + \mu)(n+1)F_n + \lambda(n+1)F_{n+1} + \mu n F_{n-1} + \mu$$

Taking the Laplace-Stieltjes transform of both sides and using the property of the LST for derivatives, we obtain

$$(\lambda + \mu + s)(n+1)L_n^* = \lambda(n+1)L_{n+1}^* + \mu n L_{n-1}^* + \mu$$

Further multiplying by z^n and summing for all n results in the following terms. Starting with the left-hand side, we have

$$\begin{aligned} (\lambda + \mu + s) \sum_{n=0}^{\infty} (n+1) L_n^* z^n &= (\lambda + \mu + s) \left[z \sum_{n=1}^{\infty} n L_n^* z^{n-1} + \sum_{n=0}^{\infty} L_n^* z^n \right] \\ &= (\lambda + \mu + s) \left(G(z) + z \frac{\partial G}{\partial z} \right) \end{aligned}$$

Proceeding with the right-hand side,

$$\begin{aligned} \text{Term 1: } \lambda \sum_{n=0}^{\infty} (n+1) L_{n+1}^* z^n &= \lambda \sum_{n=1}^{\infty} n L_n^* z^{n-1} = \lambda \frac{\partial G}{\partial z} \\ \text{Term 2: } \mu \sum_{n=0}^{\infty} n L_{n-1}^* z^n &= \mu \left[z^2 \sum_{n=2}^{\infty} (n-1) L_{n-1}^* z^{n-2} + z \sum_{n=1}^{\infty} L_{n-1}^* z^{n-1} \right] = \mu z^2 \frac{\partial G}{\partial z} + \mu z G \\ \text{Term 3: } \mu \sum_{n=0}^{\infty} z^n &= \frac{\mu}{1-z} \end{aligned}$$

After some rearrangement, we get (3.1) as stated. $\square$

Proposition 3.2.2. *The solution to (3.1) defined on $z \in B(0, 1)$ with $\Re(s) \geq 0$ is given by*

$$G(z, s) = \left[\int_{z_1}^z (x-1)^{-1} (x-z_1)^\alpha (x-z_2)^{-(\alpha+1)} dx \right] (z-z_1)^{-(\alpha+1)} (z-z_2)^\alpha \quad (3.2)$$

where z_1, z_2 are the two complex roots of the quadratic equation $\mu z^2 - (\lambda + \mu + s)z + \lambda$, z_1 being the smaller one in absolute value and $\alpha = \frac{z_1}{z_2 - z_1}$.

Proof. To solve the differential equation for $G(z, s)$, we apply the integrating factor method. In order to obtain the integrating factor, we first solve the corresponding homogeneous equation. We omit the second argument s and treat it as a fixed parameter throughout the proof.

$$G_0(z) = \frac{\lambda - (\lambda + \mu + s)z + \mu z^2}{\lambda + \mu + s - \mu z} \frac{\partial G_0}{\partial z}(z) \quad (3.3)$$

After rearranging,

$$\frac{\partial G_0}{G_0} = \frac{\lambda + \mu + s - \mu z}{\lambda - (\lambda + \mu + s)z + \mu z^2} dz \quad (3.4)$$

It is straightforward to take the integral of the left-hand side w.r.t z since its primitive function is $\ln G_0(z)$. To calculate the primitive function of the right-hand side, we use partial fraction decomposition twice.

$$\begin{aligned} \frac{(\lambda + \mu + s)}{\lambda - (\lambda + \mu + s)z + \mu z^2} &= \frac{(\lambda + \mu + s)}{\mu(z - z_1)(z - z_2)} = \frac{(\lambda + \mu + s)}{\mu(z_2 - z_1)} \left(\frac{1}{z - z_2} - \frac{1}{z - z_1} \right) \\ \frac{-\mu z}{\lambda - (\lambda + \mu + s)z + \mu z^2} &= \frac{\mu}{\mu(z_2 - z_1)} \left(\frac{z_1}{z - z_1} - \frac{z_2}{z - z_1} \right) \end{aligned}$$

The right-hand side can now be rewritten as

$$\frac{\lambda + \mu + s - \mu z}{\lambda - (\lambda + \mu + s)z + \mu z^2} = \frac{1}{\mu(z_2 - z_1)} \left[(\lambda + \mu + s - \mu z_2) \frac{1}{z - z_2} + (\mu z_1 - (\lambda + \mu + s)) \frac{1}{z - z_1} \right]$$

Therefore taking the integral w.r.t z of (3.4) gives

$$\ln G_0(z) = \frac{1}{\mu(z_2 - z_1)} [(\lambda + \mu + s - \mu z_2) \ln(z - z_2) + (\mu z_1 - (\lambda + \mu + s)) \ln(z - z_1)] + C$$

We are looking for a single solution of the homogeneous equation (to be used as the integrating factor) which means there is a free choice of the constant C , hence we proceed with $C = 0$.

Let P denote everything on the right-hand side, that is

$$P := \frac{1}{\mu(z_2 - z_1)} [(\lambda + \mu + s - \mu z_2) \ln(z - z_2) + (\mu z_1 - (\lambda + \mu + s)) \ln(z - z_1)]$$

A solution to the homogeneous equation (3.3) can be expressed as $G_0 = e^P$. Introducing the notation

$$p := \frac{\lambda + \mu + s - \mu z}{\lambda - (\lambda + \mu + s)z + \mu z^2}$$

the equation can be written as $pG_0 = G_0'$, where prime denotes differentiation w.r.t z . Plugging the solution in, we get $pe^P = P'e^P$ so that $p = P'$. The original ODE can be rewritten in terms of p as

$$G' = pG - \frac{\mu}{1 - z} \frac{1}{\mu(z - z_1)(z - z_2)}$$

Multiplying both sides by e^{-P} and using the chain rule to calculate the derivative, we have

$$(Ge^{-P})' = G'e^{-P} - Gpe^{-P} = \frac{-e^{-P}}{(1-z)(z-z_1)(z-z_2)}$$

Now taking the integral w.r.t z on both sides, Ge^{-P} is given by

$$Ge^{-P} = \int \frac{-e^{-P}}{(1-z)(z-z_1)(z-z_2)} dz + C$$

In order to express e^P and e^{-P} concisely, let

$$\alpha := \frac{\lambda + \mu + s - \mu z_2}{\mu(z_2 - z_1)} = \frac{\lambda + \mu + s - \mu z_1}{\mu(z_2 - z_1)} - \frac{\mu(z_2 - z_1)}{\mu(z_2 - z_1)} \quad (3.5)$$

and note that $z_1 + z_2 = \frac{\lambda + \mu + s}{\mu}$ so that the above expression simplifies to $\alpha = \frac{z_1}{z_2 - z_1}$. Therefore we can express e^P and e^{-P} as

$$\begin{aligned} e^P &= (z - z_2)^\alpha (z - z_1)^{-(\alpha+1)} = (z - z_2)^{\frac{z_1}{z_2 - z_1}} (z - z_1)^{\frac{-z_2}{z_2 - z_1}} \\ e^{-P} &= (z - z_1)^{\alpha+1} (z - z_2)^{-\alpha} = (z - z_1)^{\frac{z_2}{z_2 - z_1}} (z - z_2)^{\frac{-z_1}{z_2 - z_1}} \end{aligned}$$

Plugging the formulas for e^P and e^{-P} in, we get the following solution

$$G(z)(z - z_1)^{\frac{z_2}{z_2 - z_1}} (z - z_2)^{\frac{-z_1}{z_2 - z_1}} = \left(\int (z - 1)^{-1} (z - z_1)^{\frac{z_1}{z_2 - z_1}} (z - z_2)^{\frac{-z_2}{z_2 - z_1}} dz \right) + C$$

Our next task is to determine C above. In [CJMT70], the constant is determined from a boundary condition, however, in our case there is no boundary condition to use. Instead, we use an analyticity argument as seen in [HP93]. $G(z)$ is analytic in the unit disk when $\Re(s) \geq 0$, as for $s = 0$ we have

$$|L_n^*(s)| = \int_0^\infty e^{-st} f_n(t) dt = \int_0^\infty f_n(t) dt = 1$$

and for $s = a + bi, a > 0$

$$|L_n^*(s)| = \left| \int_0^\infty e^{-st} f_n(t) dt \right| \leq \int_0^\infty |e^{-at} e^{-bit}| f_n(t) dt \leq \int_0^\infty f_n(t) dt = 1$$

Hence $|G(z)| \leq \sum_{n=0}^{\infty} |L_n^*(s)| |z^n| \leq \sum_{n=0}^{\infty} |z^n| \leq \frac{1}{1-|z|}$ and consequently, $G(z)$ converges everywhere in the open unit disk. Therefore, whenever $(z - z_1)^{\frac{z_2}{z_2 - z_1}} (z - z_2)^{\frac{-z_1}{z_2 - z_1}}$ on the left-hand side has a root inside the unit disk, the right-hand side must also vanish.

In order to find a root, first note, that $|z_1| \leq \rho$ and $1 \leq |z_2|$ when $\Re(s) \geq 0$. This holds because $|z_1||z_2| = \frac{\lambda}{\mu} = \rho$ and

$$|z_1| + |z_2| \geq |z_1 + z_2| = \left| \frac{\lambda + \mu + s}{\mu} \right| \geq \frac{\lambda + \mu + \Re(s)}{\mu} \geq \rho + 1$$

Solving the resulting quadratic equation $0 \leq |z_2|^2 - (\rho + 1)|z_2| + \rho$ and recalling that $|z_1| \leq |z_2|$, we get $1 \leq |z_2|$ and $|z_1| \leq \rho$ as stated. Consequently, $(z - z_2)^{\frac{-z_1}{z_2 - z_1}}$ is a continuous function with no poles in the unit disk. The problematic z value for $(z - z_1)^{\frac{z_2}{z_2 - z_1}}$ is of course z_1 . However, x^{a+bi} can be defined in $x = 0$ continuously, provided $a > 0$ in the following way.

$$\lim_{|x| \rightarrow 0} x^{a+bi} = e^{a \ln |x|} e^{ib \ln |x|} e^{ia \arg(x)} e^{-b \arg(x)} = 0$$

Therefore, $(z - z_1)^{\frac{z_2}{z_2 - z_1}}$ has a root at z_1 provided $\Re\left(\frac{z_2}{z_2 - z_1}\right) > 0$. Applying an algebraic transformation to $(z - z_1)^{\frac{z_2}{z_2 - z_1}}$,

$$\frac{z_2}{z_2 - z_1} = \frac{\overline{z_2} - \overline{z_1}}{\overline{z_2} - \overline{z_1}} \frac{z_2}{z_2 - z_1} = \frac{\overline{z_2} z_2 - \overline{z_1} z_2}{|z_2 - z_1|^2} = \frac{1}{|z_2 - z_1|^2} \left(|z_2|^2 - z_2 \overline{z_1} \frac{z_1}{z_1} \right) = \frac{|z_2|^2 - \rho \frac{\overline{z_1}}{z_1}}{|z_2 - z_1|^2}$$

the real part of $\frac{z_2}{z_2 - z_1}$ can be shown to be positive as follows

$$\Re\left(\frac{z_2}{z_2 - z_1}\right) = \frac{1}{|z_2 - z_1|^2} \left(|z_2|^2 - \rho \Re\left(\frac{\overline{z_1}}{z_1}\right) \right) > \frac{1}{|z_2 - z_1|^2} (\rho - \rho 1) = 0$$

where we used that $-1 \leq \Re\left(\frac{\overline{z_1}}{z_1}\right) \leq 1$ and $|z_2|^2 > \rho$.

From the above, $(z_1 - z_1)^{\frac{z_2}{z_2 - z_1}} = 0$ and $\left| (z_1 - z_2)^{\frac{-z_1}{z_2 - z_1}} \right| < \infty$ when $\Re(s) \geq 0$, and the expression $(z - z_1)^{\frac{z_2}{z_2 - z_1}} (z - z_2)^{\frac{-z_1}{z_2 - z_1}}$ has a root at $z = z_1$. Consequently, we can use a definite integral instead of an indefinite one in (3.2), eliminating the unknown C .

$$G(z)(z - z_1)^{\frac{z_2}{z_2 - z_1}} (z - z_2)^{\frac{-z_1}{z_2 - z_1}} = \int_{z_1}^z (x - 1)^{-1} (x - z_1)^{\frac{z_1}{z_2 - z_1}} (x - z_2)^{\frac{-z_2}{z_2 - z_1}} dx$$

So that the final solution, using the α -notation from (3.5), is

$$G(z) = \left(\int_{z_1}^z (x-1)^{-1} (x-z_1)^\alpha (x-z_2)^{-(\alpha+1)} dx \right) (z-z_1)^{-(\alpha+1)} (z-z_2)^\alpha$$

Our final task is to show that the expression for $G(z)$ above is analytic everywhere inside the unit disk. The only problematic point is z_1 due to the $(x-z_1)^\alpha$ term of the integrand and the $(z-z_1)^{-(\alpha+1)}$ multiplier. These functions have exponents with potentially negative real parts; therefore, when studied in isolation, they are only well-defined if an appropriate branch cut is added to prevent going around z_1 in the unit disk.

Let us examine $G(z)$ in the neighbourhood of z_1 . The other two terms of the integrand are holomorphic inside the unit disk, therefore they can be expressed in a power series form centred around z_1 as $h(x) = \sum_{n=0}^{\infty} a_n (x-z_1)^n$. The integral in $G(z)$ can be written as

$$\begin{aligned} I(x) &= \int h(x) (x-z_1)^\alpha dx = \int \sum_{n=0}^{\infty} a_n (x-z_1)^{n+\alpha} dx = \sum_{n=0}^{\infty} a_n \int (x-z_1)^{n+\alpha} dx \\ &= (x-z_1)^{\alpha+1} \sum_{n=0}^{\infty} \frac{a_n}{n+\alpha+1} (x-z_1)^n = (x-z_1)^{\alpha+1} H(x) \end{aligned}$$

where $H(x)$ is analytic near z_1 . Note that $I(x)$ does not have a pole in z_1 as $\Re(\alpha+1) > 0$, hence $I(z_1) = 0$. Therefore, $G(z)$ in the neighbourhood of z_1 can be expressed as

$$G(z) = I(z) (z-z_1)^{-(\alpha+1)} (z-z_2)^\alpha = H(z) (z-z_2)^\alpha$$

This shows that $G(z)$ preserves analyticity near z_1 , and is therefore analytic everywhere in the unit disk which concludes the proof. $\square$

We know that at $s = 0$, $G(z, 0) = \frac{1}{1-z}$, as at $s = 0$ all the coefficients evaluate to 1. We show that taking the limit $|s| \rightarrow 0$, the expression obtained for $G(z, s)$ in Proposition 3.2.2 evaluates to $\frac{1}{1-z}$ as expected. Let us first look at how the roots z_1 and z_2 behave when $|s|$ tends to 0.

$$\lim_{|s| \rightarrow 0} z_1 = \lim_{|s| \rightarrow 0} \frac{\lambda + \mu - \sqrt{(\lambda + \mu)^2 - 4\lambda\mu}}{2\mu} = \rho \qquad \lim_{|s| \rightarrow 0} z_2 = \lim_{|s| \rightarrow 0} \frac{\rho}{z_1} = 1$$

Next, we study the behaviour of the integral part of the formula when s goes to zero.

Proposition 3.2.3. *Assume z_1 and z_2 are defined as before, then*

$$\lim_{|s| \rightarrow 0} \int_{z_1}^z (x-1)^{-1} (x-z_1)^{\frac{z_1}{z_2-z_1}} (x-z_2)^{\frac{-z_2}{z_2-z_1}} dx = \int_{\rho}^z (x-1)^{-1} (x-\rho)^{\frac{\rho}{1-\rho}} (x-1)^{\frac{-1}{1-\rho}} dx$$

Proof. We first transform the integral so that the boundary is not dependent on s . Using integration by substitution, if we denote the original curve $F \circ \gamma := [z_1, z]$ and define $F^{-1}(x) := \frac{x-z_1}{z-z_1}z$ then $\gamma = [0, z]$. F is then $F(x) = \frac{z-z_1}{z}x + z_1$ and the derivative is $F'(x) = \frac{z-z_1}{z}$.

$$\begin{aligned} \lim_{|s| \rightarrow 0} \int_{z_1}^z (x-1)^{-1} (x-z_1)^{\frac{z_1}{z_2-z_1}} (x-z_2)^{\frac{-z_2}{z_2-z_1}} dx = \\ \lim_{|s| \rightarrow 0} \int_0^z \left(\frac{z-z_1}{z}x + z_1 - 1 \right)^{-1} \left(\frac{z-z_1}{z}x \right)^{\frac{z_1}{z_2-z_1}} \left(\frac{z-z_1}{z}x + z_1 - z_2 \right)^{\frac{-z_2}{z_2-z_1}} \left(\frac{z-z_1}{z} \right) dx \end{aligned}$$

The following three lemmas help break up the problem, the proofs are straightforward and presented in Appendix A.1.1 for completeness. Let f_n, f, g_n, g be complex functions and γ a rectifiable curve.

Lemma 3.2.1. *If $f_n \rightarrow f$ uniformly along γ then $\int_{\gamma} f_n \rightarrow \int_{\gamma} f$.*

Lemma 3.2.2. *If $f_n \rightarrow f$ uniformly, $g_n \rightarrow g$ uniformly, furthermore f and g are bounded, then $f_n g_n \rightarrow f g$ uniformly.*

Lemma 3.2.3. *If $f_n \rightarrow f$ uniformly, $\exists B$ compact set for which $\exists N_B : n > N_B \text{ Im}(f_n) \subseteq B$ and g continuous on at least B then $g \circ f_n \rightarrow g \circ f$ uniformly.*

Based on Lemma 3.2.1 it is enough to prove the function inside the integral converges uniformly along $[0, z]$ and based on Lemma 3.2.2, as long as the limiting functions are bounded, we can break up the function even further and consider each bracketed part on its own when proving uniform convergence.

Part 1: Let $p_n(x) = \left(\frac{z-z_1^{(n)}}{z}x + z_1^{(n)} - 1 \right)^{-1}$ and $p(x) = \left(\frac{z-\rho}{z}x + \rho - 1 \right)^{-1}$ then $p_n(x) \rightarrow p(x)$ when $s \rightarrow 0$ uniformly in x along $[0, z]$, where $z_1^{(n)}$ denotes the series of z_1 values when $s \rightarrow 0$.

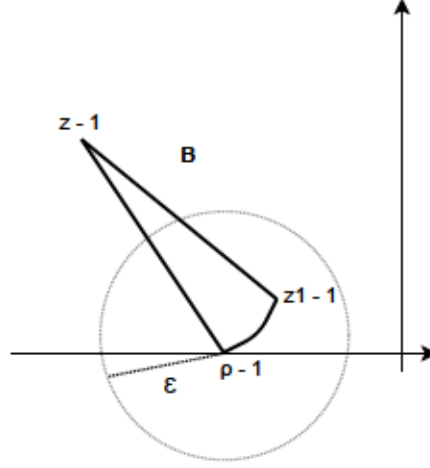


Figure 3.1: Compact set B containing $\text{Im}(f_n) \subseteq B$ when n is "large"

Using Lemma 3.2.3, first we need to check that $f_n(x) = \frac{z-z_1^{(n)}}{z}x + z_1^{(n)} - 1$ converges uniformly to $f(x) = \frac{z-\rho}{z}x + \rho - 1$ along $[0, z]$.

$$\begin{aligned} |f_n(x) - f(x)| &= \left| \frac{z-z_1^{(n)}}{z}x + z_1^{(n)} - 1 - \left(\frac{z-\rho}{z}x + \rho - 1 \right) \right| = \left| \frac{\rho - z_1^{(n)}}{z}x + z_1^{(n)} - \rho \right| \\ &\leq |\rho - z_1^{(n)}| \left| \frac{x}{z} - 1 \right| \leq \epsilon M \end{aligned}$$

where the last inequality holds as $[0, z]$ is a closed interval, therefore on $[0, z]$, $\left| \frac{x}{z} - 1 \right| < M$ for some M .

Furthermore, $g(x) = 1/x$ is a continuous function on $\mathbb{C} \setminus \{0\}$. Therefore, we need to find a compact set B that contains the image of f_n for large n values but does not contain 0.

Note, when $x = 0$: $f_n(0) = z_1 - 1$ and $f(0) = \rho - 1$, and when $x = z$: $f_n(z) = f(z) = z - 1$. All three of those points have negative real parts since $|z| < 1$, $|z_1| < \rho$ and $\rho < 1$. Furthermore, if $n > N_\epsilon$: $|\rho - z_1^{(n)}| < \epsilon$, so all the $(z_1^{(n)} - 1)$ values lie within ϵ distance from $(\rho - 1)$. If ϵ is sufficiently small, these have negative real parts as well.

Let $B := \{f_n(x) | x \in [0, z], n > N_\epsilon\} \cup \{f(x) | x \in [0, z]\}$, this is a bounded and closed set, hence it is compact and $\{0\} \notin B$. Set B is illustrated in Figure 3.1.

All the conditions of Lemma 3.2.3 are satisfied, therefore $p_n(x) \rightarrow p(x)$ uniformly.

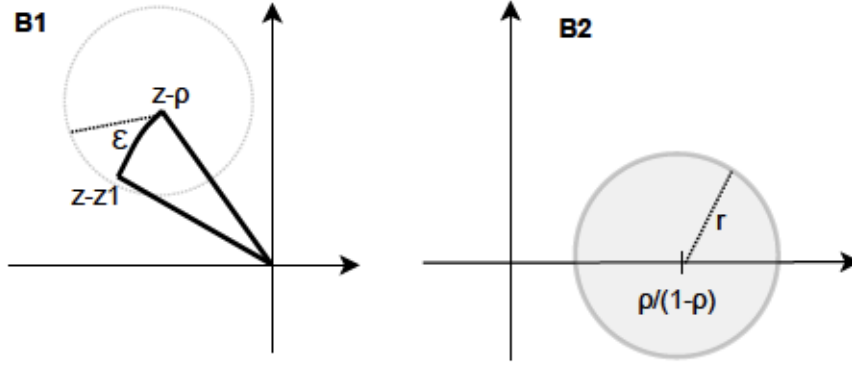


Figure 3.2: Compact sets $B1 : \text{Im}(f_n) \subseteq B1$ and $B2 : \text{Im}(g_n) \subseteq B2$

Part 2: Let $q_n(x) = \left(\frac{z-z_1^{(n)}}{z} x \right)^{\frac{z_1^{(n)}}{z_2^{(n)}-z_1^{(n)}}}$ and $q(x) = \left(\frac{z-\rho}{z} x \right)^{\frac{\rho}{1-\rho}}$ then $q_n(x) \rightarrow q(x)$ when $s \rightarrow 0$ uniformly along $x \in [0, z]$.

Due to the exponent being dependent on s , we cannot proceed exactly as before. Defining

$$f_n(x) := \frac{z - z_1^{(n)}}{z} x, \quad g_n(x) := \frac{z_1^{(n)}}{z_2^{(n)} - z_1^{(n)}} \text{ so that } q_n(x) = f_n(x)^{g_n(x)}$$

$$f(x) := \frac{z - \rho}{z} x, \quad g(x) := \frac{\rho}{1 - \rho} \text{ so that } q(x) = f(x)^{g(x)}$$

We start by defining two compact sets in the complex plane, shown in Figure 3.2

$$B1 := \{f_n(x) | x \in [0, z]\} \cup \{f(x) | x \in [0, z]\}$$

$$B2 := \overline{B} \left(\frac{\rho}{1 - \rho}, r \right), \quad r < \frac{\rho}{1 - \rho}$$

$\text{Im}(f_n) \subseteq B1$ and $\text{Im}(g_n) \subseteq B2$ by construction.

Lemma 3.2.4. *The function $F(x, y) := x^y$ is uniformly continuous on $B1 \times B2$, that is*

$$\forall \epsilon \exists \delta_\epsilon \text{ so that } \forall (x_0, y_0), (x_1, y_1) \in B1 \times B2 : \|(x_1, y_1) - (x_0, y_0)\| < \delta_\epsilon \Rightarrow |x_1^{y_1} - x_0^{y_0}| < \epsilon.$$

The proof is provided in Appendix A.1.1. In order to prove uniform convergence of q_n to q , we

need a threshold ξ_ϵ not dependent on $x \in [0, z]$, so that if $|s| < \xi_\epsilon$,

$$\left| \left(\frac{z - z_1^{(n)}}{z} x \right)^{\frac{z_1^{(n)}}{z_2^{(n)} - z_1^{(n)}}} - \left(\frac{z - \rho}{z} x \right)^{\frac{\rho}{1-\rho}} \right| < \epsilon \quad (3.6)$$

Recall that $z_1^{(n)}$ and $z_2^{(n)}$ denote the series of z_1, z_2 values as $s \rightarrow 0$, respectively. Instead of finding threshold ξ_ϵ , we can equivalently find N_ϵ for which $n > N_\epsilon$ ensures that (3.6) holds.

We start by fixing x . From Lemma 3.2.4, we know that if $\left| \left(\frac{z - z_1^{(n)}}{z} x \right) - \left(\frac{z - \rho}{z} x \right) \right| < \delta_\epsilon/2$ and $\left| \frac{z_1^{(n)}}{z_2^{(n)} - z_1^{(n)}} - \frac{\rho}{1-\rho} \right| < \delta_\epsilon/2$, then (3.6) holds. Therefore, we find N_ϵ for which $n > N_\epsilon$ guarantees the above two inequalities.

$$\left| \frac{z - z_1^{(n)}}{z} x - \frac{z - \rho}{z} x \right| = \frac{|x|}{|z|} |\rho - z_1^{(n)}| \leq |\rho - z_1^{(n)}| < \delta_\epsilon/2 \text{ if } n > N_1$$

$$\left| \frac{z_1^{(n)}}{z_2^{(n)} - z_1^{(n)}} - \frac{\rho}{1-\rho} \right| < \delta_\epsilon/2 \text{ if } n > N_2$$

Let $N_\epsilon := \max(N_1, N_2)$ and note that N_ϵ does not depend on x . Therefore, $\forall x \in [0, z]$ we have

$$\left\| \left(\frac{z - z_1^{(n)}}{z} x, \frac{z_1^{(n)}}{z_2^{(n)} - z_1^{(n)}} \right) - \left(\frac{z - \rho}{z} x, \frac{\rho}{1-\rho} \right) \right\| < \delta_\epsilon/2 + \delta_\epsilon/2 = \delta_\epsilon$$

so that by Lemma 3.2.4, (3.6) holds $\forall x \in [0, z]$ as well.

Part 3: Let $r_n(x) = \left(\frac{z - z_1^{(n)}}{z} x + z_1^{(n)} - z_2^{(n)} \right)^{\frac{-z_2^{(n)}}{z_2^{(n)} - z_1^{(n)}}}$ and $r(x) = \left(\frac{z - \rho}{z} x + \rho - 1 \right)^{\frac{-1}{1-\rho}}$ then $r_n(x) \rightarrow r(x)$ when $s \rightarrow 0$ uniformly along $x \in [0, z]$.

This can be proved analogously to Part 2, therefore it is omitted here for brevity. The details are included in Appendix A.1.1.

Part 4: Here, we need $s_n(x) = \frac{z - z_1}{z} \rightarrow s(x) = \frac{z - \rho}{z}$ uniformly in x which is trivially true.

Finally, the limiting functions

$$p(x) = \left(\frac{z - \rho}{z} x + \rho - 1 \right)^{-1} \quad r(x) = \left(\frac{z - \rho}{z} x + \rho - 1 \right)^{\frac{-1}{1-\rho}}$$

$$q(x) = \left(\frac{z - \rho}{z} x \right)^{\frac{\rho}{1-\rho}} \quad s(x) = \frac{z - \rho}{z}$$

are all bounded on $x \in [0, z]$, consequently due to Lemma 3.2.2 the integrand converges uniformly. This concludes the proof of Proposition 3.2.3. $\square$

We are now ready to study the behaviour of $G(z, s)$ when s goes to 0.

Proposition 3.2.4. *Let $G(z, s)$ be defined as in (3.2), $z \in B(0, 1)$ and $\Re(s) \geq 0$, then*

$$\lim_{|s| \rightarrow 0} G(z, s) = \frac{1}{1 - z}$$

Proof. Using Proposition (3.2.3),

$$\lim_{|s| \rightarrow 0} G(z, s) = \left[\int_{\rho}^z (x - 1)^{-1} (x - \rho)^{\frac{\rho}{1-\rho}} (x - 1)^{\frac{-1}{1-\rho}} dx \right] (z - \rho)^{\frac{-1}{1-\rho}} (z - 1)^{\frac{\rho}{1-\rho}}$$

Working backwards,

$$\begin{aligned} \left[\int_{\rho}^z (x - 1)^{-1} (x - \rho)^{\frac{\rho}{1-\rho}} (x - 1)^{\frac{-1}{1-\rho}} dx \right] (z - \rho)^{\frac{-1}{1-\rho}} (z - 1)^{\frac{\rho}{1-\rho}} &= \frac{1}{1 - z} \Leftrightarrow \\ \int_{\rho}^z (x - 1)^{-1} (x - \rho)^{\frac{\rho}{1-\rho}} (x - 1)^{\frac{-1}{1-\rho}} dx &= -(z - 1)^{\frac{-1}{1-\rho}} (z - \rho)^{\frac{1}{1-\rho}} \Leftrightarrow \\ (z - 1)^{-1} (z - \rho)^{\frac{\rho}{1-\rho}} (z - 1)^{\frac{-1}{1-\rho}} &= \frac{\partial}{\partial z} (-(z - 1)^{\frac{-1}{1-\rho}} (z - \rho)^{\frac{1}{1-\rho}}) \end{aligned}$$

To check the last line holds (using the notation $\beta = \frac{\rho}{1-\rho}$ for readability),

$$\begin{aligned} &\frac{\partial}{\partial z} (-(z - 1)^{-(\beta+1)} (z - \rho)^{(\beta+1)}) \\ &= (\beta + 1)(z - 1)^{-(\beta+2)} (z - \rho)^{(\beta+1)} - (z - 1)^{-(\beta+1)} (\beta + 1)(z - \rho)^{\beta} \\ &= (z - 1)^{-(\beta+2)} (z - \rho)^{\beta} [(\beta + 1)(z - \rho) - (\beta + 1)(z - 1)] \\ &= (z - 1)^{-(\beta+2)} (z - \rho)^{\beta} [(\beta + 1)(1 - \rho)] \end{aligned}$$

Plugging in the expression for β , we get

$$(z-1)^{\frac{p-2}{1-p}}(z-\rho)^{\frac{p}{1-p}} \left[\frac{1}{1-\rho}(1-\rho) \right] = (z-1)^{-1}(z-1)^{\frac{-1}{1-p}}(z-\rho)^{\frac{p}{1-p}}$$

which concludes the proof. $\square$

Using (3.2), the LST of the response time distribution can be expressed as

$$W(s) = (1-\rho) \left[\int_{z_1}^{\rho} (x-1)^{-1}(x-z_1)^{\alpha}(x-z_2)^{-(\alpha+1)} dx \right] (\rho-z_1)^{-(\alpha+1)}(\rho-z_2)^{\alpha} \quad (3.7)$$

where z_1, z_2 are the two complex roots of the quadratic equation $\mu z^2 - (\lambda + \mu + s)z + \lambda$, z_1 being the smaller one in absolute value and $\alpha = \frac{z_1}{z_2 - z_1}$. Comparing our alternative solution (3.7) with the original (2.11) derived in [CJMT70], the main difference is that our expression gives the LST in a finite integral form, whereas the original contains an improper integral with an infinite upper limit. [Mor85] shows how to transform this improper integral and invert it analytically to express the response time distribution in a finite integral form; however, to reuse the result as a building block — for example in a tandem network of two PS queues — and to perform steps analogous to those in [HB21b], where the LST of the response-time distribution is computed for a tandem network of FCFS queues with batch processing, the unconditional LST is required in a finite form. In terms of computational complexity, calculating a finite integral should, in theory, be less expensive than evaluating an improper one. However, in practice, both formulas are evaluated instantaneously in Mathematica.

3.2.1 Numerical Results

We perform the deconditioning of the conditional LST obtained in [CJMT70] and compare it to the LST obtained in this chapter. The numerical experiments have been implemented in Mathematica. To check the correctness of the calculated LST, we plot the functions along various inputs for light and heavy loads.

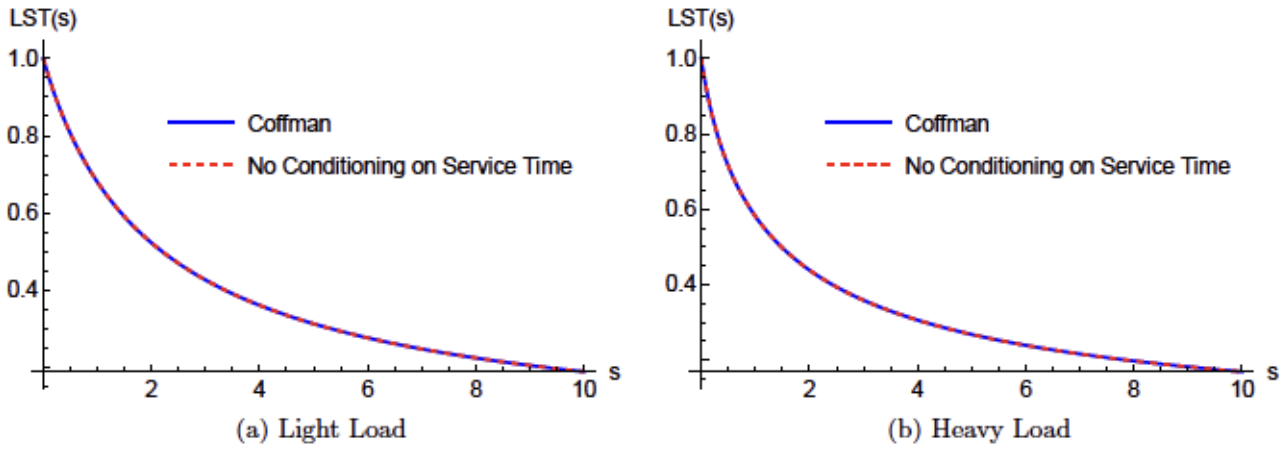


Figure 3.3: LST along the positive reals in an M/M/1-PS queue with parameters $\lambda = 1, \mu = 3$ (Light Load) and $\lambda = 9, \mu = 10$ (Heavy Load).

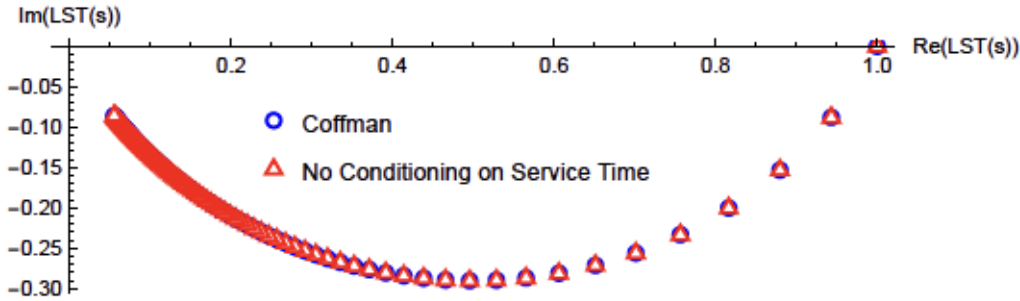


Figure 3.4: LST in the first quadrant in an M/M/1-PS queue with parameters $\lambda = 1$ and $\mu = 3$

Figure 3.3a and Figure 3.3b show the LSTs along the real axis. For positive real inputs, the LST is also real, which allows us to plot $s \mapsto LST(s)$ in 2 dimensions. Figure 3.4 and Figure 3.5 show the function values in the first quadrant along the line $s + 2si$, in this case, both the inputs and outputs are complex, hence we only plot the latter. In all the figures above the values match exactly as far as one can tell by eyeballing. In order to better compare the LSTs, Table 3.1 and 3.2 display the function values in the second quadrant along the line $s - si$. The difference is zero up to numerical precision for all inputs in the tables. Finally, we use the Fixed Talbot inversion algorithm¹ [AW95] to invert the LST calculated by the no-conditioning method for both the light and heavy loads, shown in Figure 3.6. Unsurprisingly, a heavy load has a heavier tale compared to a light load. As expected, the Fixed Talbot method was unable to invert the LST in the case of Coffman's formula due to numerical instability for certain

¹See <https://mpmath.org/doc/current/calculus/inverselaplace.html>.

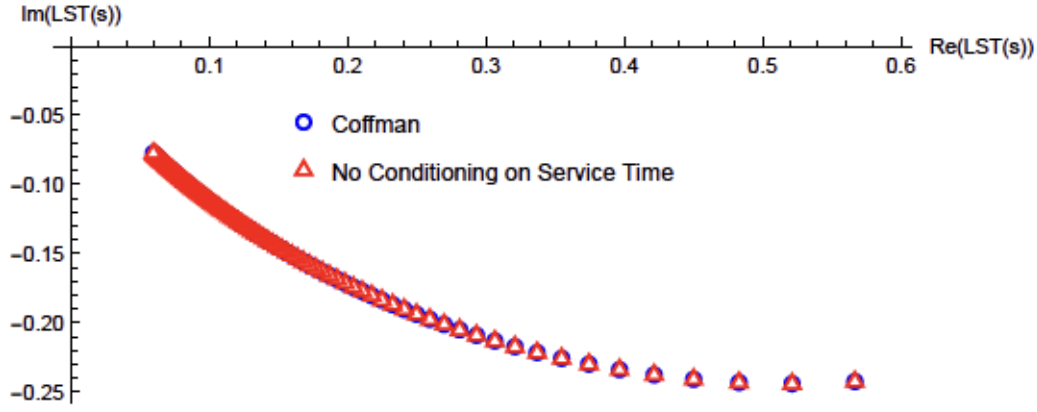


Figure 3.5: LST in the first quadrant in an $M/M/1$ -PS queue with parameters $\lambda = 9$ and $\mu = 10$

Table 3.1: LST in the fourth quadrant in an $M/M/1$ -PS queue with parameters $\lambda = 1$ and $\mu = 3$

Input	LST - No Conditioning	Coffman's method	Diff
10000 – 10000i	0.00012164 + 0.000121604i	0.00012164 + 0.000121604i	0
20000 – 20000i	0.0000608198 + 0.0000608108i	0.0000608198 + 0.0000608108i	0
30000 – 30000i	0.0000405465 + 0.0000405425i	0.0000405465 + 0.0000405425i	0
40000 – 40000i	0.0000304099 + 0.0000304077i	0.0000304099 + 0.0000304077i	0
50000 – 50000i	0.0000243279 + 0.0000243265i	0.0000243279 + 0.0000243265i	0
60000 – 60000i	0.0000202733 + 0.0000202723i	0.0000202733 + 0.0000202723i	0

values of s ; specifically, the improper integral fails to converge because the integrand depends on near-cancellation between the numerator and denominator.

3.3 $M/M/1$ -PS: approximate solution

In this section, we implement the method detailed in [Har24] for solving functional equations to find a solution for (3.1) in Proposition 3.2.1. We have already presented two exact solutions for calculating the response time density in an $M/M/1$ -PS queue, both of which are highly accurate and computationally efficient. This third solution, while less accurate and slower, is not intended as yet another method for solving the same queue. Rather, we use this problem, for which an exact solution is known, to evaluate the numerical approximation method and to provide a detailed analysis of its accuracy, as preparation for Section 3.4 and Chapter 4, where the method is applied to compute the response time density in two previously unsolved queueing networks. The method is first explained via an introductory example in Section

Table 3.2: LST in the fourth quadrant in an M/M/1-PS queue with parameters $\lambda = 9$ and $\mu = 10$

Input	LST - No Conditioning	Coffman's method	Diff
10000 – 10000i	0.000127921 + 0.000127834i	0.000127921 + 0.000127834i	0
20000 – 20000i	0.0000639607 + 0.0000639389i	0.0000639607 + 0.0000639389i	0
30000 – 30000i	0.0000426405 + 0.0000426308i	0.0000426405 + 0.0000426308i	0
40000 – 40000i	0.0000319803 + 0.0000319749i	0.0000319803 + 0.0000319749i	0
50000 – 50000i	0.0000255843 + 0.0000255808i	0.0000255843 + 0.0000255808i	0
60000 – 60000i	0.0000213202 + 0.0000213178i	0.0000213202 + 0.0000213178i	0

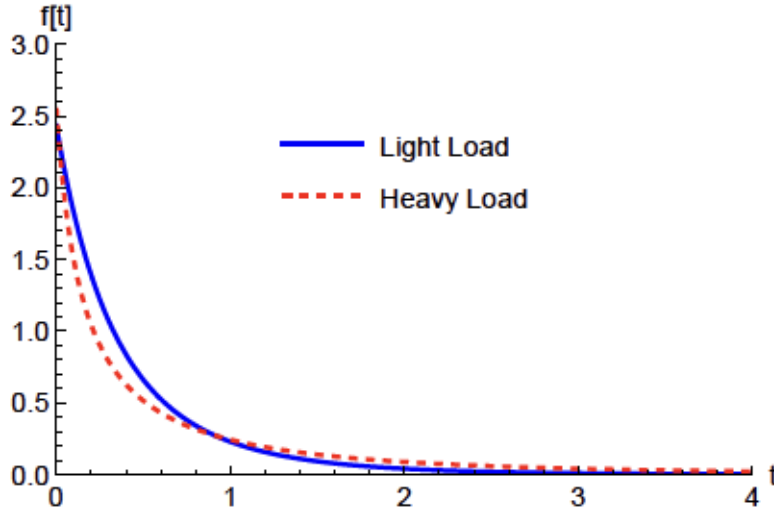


Figure 3.6: Density function of the response time in an M/M/1-PS queue with parameters $\lambda = 1, \mu = 3$ (Light Load) and $\lambda = 9, \mu = 10$ (Heavy Load).

3.3.1. In Section 3.3.2, the approximation method is applied to solve the original equation. Next, an optimisation method is proposed and a recursive formula for efficient calculation of the response time moments is derived. Finally, an extensive evaluation of the approximation method at multiple stages of the calculations is presented in Section 3.3.3.

3.3.1 Introductory example

Consider the equation $f'(z) = 2z$ for $z \in \mathbb{C}, |z| \leq 1$, with boundary condition $f(0) = 0$. We want to find a good approximation of $f(z)$ everywhere on a closed unit disk. Note, however, that knowing $f(z)$ at a sufficiently large finite number of different values on the boundary of the disk is sufficient to get a good approximation of $f(z)$ *everywhere* inside it. This can be achieved in three simple steps as shown in Figure 3.7. Working backwards, due to Cauchy's Integral Theorem 2.8, if we knew $f(z)$ for any $|z| = 1$, that is on the boundary of the disk centred at

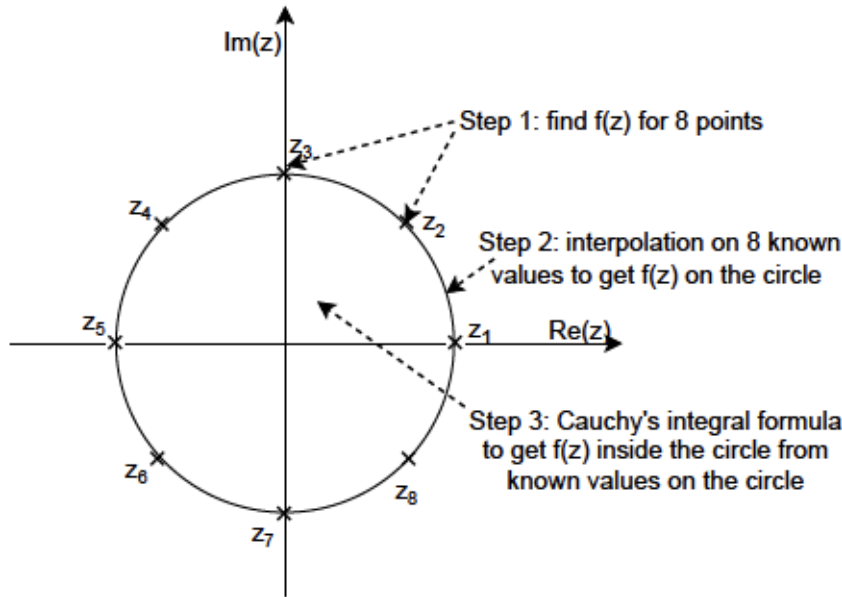


Figure 3.7: Steps to approximate $f(z)$ from 8 known values around the unit circle.

the origin with radius 1, we could calculate $f(z)$ for any z with $|z| < 1$ inside the boundary by evaluating an appropriate Cauchy integral (Step 3). Therefore, our updated goal is to solve $f(z)$ for $|z| = 1$. Now, instead of solving for all $|z| = 1$, we pick $n = 8$ equidistant points around the unit circle; let these be $\vec{z} = \{z_1, \dots, z_8\} = \{e^{ik\pi/4} | k = 0, \dots, 7\}$. We aim to solve the equation for just these 8 points. Once $\vec{f} = \{f(z_1), \dots, f(z_8)\}$ is known, f is approximated on the boundary by interpolating the 8 known values around the circle (Step 2). Choosing the boundary to be a circle means the interpolation can be done easily on the real-valued arguments (angles) of the discretisation points, giving more precision. Moreover, interpolation on reals is directly supported in most mathematical software packages. We found linear interpolation to be sufficient in our numerical experiments, producing very accurate results; however, higher-order methods could also be employed.

In order to find the values of f at the selected 8 points around the unit circle, or in other words to solve for the vector $\vec{f}$, the functional equation is transformed into a set of 8 linear equations. Since we are solving for a finite set of input values, a linear approximation of all the operators present in the equation for $f(z)$ naturally leads to matrix representations. We refer to these as the operators' *matrix representations*, allowing us to rewrite the functional equations

as a matrix-vector equation. In this introductory example, the only operation is taking the first derivative. The matrix representations of various operations are given in [Har24] and the table corresponding to a single variable is included in the Appendix as Table 1. Looking up differentiation in the table, we find that the matrix needed for our introductory example is $\mathbf{M}_1$. In [Har24], $\mathbf{M}_1$ is given for an arbitrary number of points; however, once the number of these is fixed (in this case to $n = 8$), $\mathbf{M}_1 \in \mathbb{C}^{8 \times 8}$ becomes a constant matrix. That is to say, $\mathbf{M}_1$ does not depend on $f(z)$, meaning that for any analytic function $g(z)$ given at 8 points around a contour in the complex plane – denoting its value at these given points as a vector $\vec{g} \in \mathbb{C}^{8 \times 8}$ – we have $\vec{g}^{(1)} \approx \mathbf{M}_1 \vec{g}$, where $\vec{g}^{(1)}$ is the first derivative of $g(z)$ evaluated at the same 8 input values as $\vec{g}$.

Hence, the equation $f'(z) = 2z$ is transformed into $\mathbf{M}_1 \vec{f} = 2\vec{z}$. However, $f'(z) = 2z$ has a degree of freedom which means that $\text{rank}(\mathbf{M}_1) = 7$. The unique solution is given by the boundary condition, $f(0) = 0$, which provides the 8th, linearly independent equation. Again, we use Table 1 to look up the matrix representing the calculation of $f(z)$ at an internal point. The corresponding matrix is $\mathbf{K}_0^{(0)}$ which has identical rows and approximates the vector $\{f(0), \dots, f(0)\}^\top \approx \mathbf{K}_0^{(0)} \vec{f}$. Let $\mathbf{A}$ be the matrix we get from $\mathbf{M}_1$ by replacing its last row with the first row (for example) of $\mathbf{K}_0^{(0)}$. Furthermore let $\vec{r} = (2z_1, \dots, 2z_7, 0)$. $\mathbf{A}$ is full rank, therefore $\vec{f} = \mathbf{A}^{-1} \vec{r}$.

The matrices $\mathbf{M}_1$ and $\mathbf{A}^{-1}$ are given explicitly in Appendix A.1.2. The numerics of this introductory example are shown in Table 3.3. The first column displays the 8 discretisation points; the second column is the solution for the unknown values $\vec{f}$ calculated by the above method; this is compared against the exact known solution $f(z) = z^2$ in the third and fourth columns. The error is around 1% for each point.

Above, we set the number of discretisation points to $n = 8$, which is relatively small. This choice was made so that Table 3.3 and the matrices, $\mathbf{M}_1$ and $\mathbf{A}^{-1}$, used in the calculation could be displayed in full. In general, there is a trade-off between accuracy and computational complexity: increasing the number of discretisation points around the circle improves the accuracy of the matrix representations of the operators, and consequently, the accuracy of the

Table 3.3: Numerical results for solving the introductory example $f'(z) = 2z$ with boundary condition $f(0) = 0$ using $n = 8$ discretisation points around the unit circle.

$\bar{z}$	$\vec{f} = \mathbf{A}^{-1}\vec{r}$	$\bar{z}^2$	$ \vec{f} - \bar{z}^2 $
1	0.99	1	0.01
$e^{i\pi/4}$	0.99i	i	0.01
i	-0.99	-1	0.01
$e^{i3\pi/4}$	-0.99i	-i	0.01
-1	0.99	1	0.01
$e^{i5\pi/4}$	0.99i	i	0.01
-i	-0.99	-1	0.01
$e^{i7\pi/4}$	-0.99i	-i	0.01

solution vector in Step 1 of Figure 3.7. Moreover, knowing f at more points around the circle increases the accuracy of the interpolation step as well. In our case, f is a very smooth function, so even $n = 8$ points provide a reasonable estimate. However, for less smooth functions, $n = 8$ would be far too few. On the other hand, n directly controls the size of the matrices that must be computed. In one dimension, the matrices are of size $n \times n$, but when applying the method to queueing networks, the problem becomes two-dimensional, and the matrix sizes grow to $n^2 \times n^2$. Furthermore, the more complex the functional equation, the greater the number of operations requiring a matrix representation. In Section 4.4.4, we present a formal analysis of the worst-case error of the approximation method, along with its time and space complexity.

It is important to note that while [Har24] outlines the theoretical foundation for computing the matrices in Table 1, it does not include the corresponding code for their generation. Implementing this method is challenging due to the complexity of the formulas that define these matrices.

3.3.2 Solving the functional equation

We follow the same steps to solve the functional equations for the generating function of the LST of response time densities in an M/M/1-PS queue. Using the notation of Table 1 and the method described above, (3.1) becomes

$$(\Lambda(\vec{a}_1)\mathbf{M}_1 + \Lambda(\vec{a}_2))\vec{g} = \vec{a}_3$$

where $\vec{a}_1, \vec{a}_2, \vec{a}_3$ are the vector representations of the functions A_1, A_2, A_3 defined by

$$A_1(z) = -\lambda + (\lambda + \mu + s)z - \mu z^2$$

$$A_2(z) = \lambda + \mu + s - \mu z$$

$$A_3(z) = \frac{\mu}{1-z}$$

Defining $\Gamma_G(s)$ as

$$\Gamma_G(s) := (\Lambda(\vec{a}_1)M_1 + \Lambda(\vec{a}_2))^{-1}$$

the solution for $\vec{g}$ can be obtained as

$$\vec{g} = \Gamma_G(s)\vec{a}_3$$

Note that as the notation suggests, $\Gamma_G(s)$ depends on the value of the Laplace parameter s , therefore it needs to be recomputed for each s value used during the inversion of the LST or the calculation of the moments.

An optimisation

Next, we note that as seen in Proposition 3.2.4, when $s = 0$ the generating function $G(z, s) = \frac{1}{1-z}$, that is G has a pole at $z = 1$. We define a new function

$$H(z, s) := (1-z)G(z, s)$$

for which $H(z, 0) = 1$. This constant function is numerically more desirable than $G(z, s)$ with its sharp peak near $z = 1$. The LST can be expressed in terms of $H(z, s)$ as

$$W(s) = H(\rho, s)$$

We now obtain a functional equation for H by multiplying both sides of (3.1) by $1 - z$ and using the definition of H to express the G terms. The new functional equation is

$$\left(\mu + \frac{s}{1-z}\right)H(z, s) - (\lambda - (\lambda + \mu + s)z + \mu z^2)\frac{\partial H}{\partial z}(z, s) = \mu \quad (3.8)$$

and the corresponding matrix-vector equation is

$$\left(\Lambda(\vec{b}_1) - \Lambda(\vec{b}_2)M_1\right)\vec{h} = (\mu, \dots, \mu)^\top$$

where $\vec{b}_1, \vec{b}_2$ are the vector representations of the functions B_1, B_2 defined by

$$\begin{aligned} B_1(z) &= \mu + \frac{s}{1-z} \\ B_2(z) &= \lambda - (\lambda + \mu + s)z + \mu z^2 \end{aligned}$$

Define $\Gamma_H(s)$ as

$$\Gamma_H(s) := \left(\Lambda(\vec{b}_1) - \Lambda(\vec{b}_2)M_1\right)^{-1} \quad (3.9)$$

the solution for $\vec{h}$ can be obtained as

$$\vec{h} = \Gamma_H(s)(\mu, \dots, \mu)^\top$$

Calculating the moments

The moments of the response time distribution can be calculated from the LST using the following formula. Let m_n denote the n -th moment, then

$$m_n = (-1)^n \frac{\partial^n}{\partial s^n} W(0) = (-1)^n \frac{\partial^n}{\partial s^n} H(\rho, 0)$$

In the following lemma, we derive a recursive formula for calculating derivatives of $H(z, s)$ with respect to s using its vector form $\vec{h}$.

Lemma 3.3.1. *The k th partial derivative of $H(z, s)$ with respect to s at $s = 0$, has vector representation $\vec{h}_k$ given by the recurrence*

$$\begin{aligned}\vec{h}_0 &= (1, \dots, 1)^\top \\ \vec{h}_k &= \Gamma_H(0) \left(-k\Lambda(\vec{c}) \vec{h}_{k-1} - k\Lambda(\vec{z}) \vec{h}'_{k-1} \right) \text{ for } k \geq 1 \\ \vec{h}'_k &= M_1 \vec{h}_k\end{aligned}$$

where $\Gamma_H(0)$ is defined by (3.9) with substitution $s = 0$ and $\vec{c}$ is the vector representation of function C defined as $C(z) = \frac{1}{1-z}$.

Proof. First note that $H(z, 0) = 1$ by construction. Differentiating (3.8) k times w.r.t s using Leibnitz's result for multiple differentiation of a product, for $k \geq 1$

$$\begin{aligned}\left(\mu + \frac{s}{1-z} \right) H^{(k)}(z, s) - (\lambda - (\lambda + \mu + s)z + \mu z^2) \frac{\partial}{\partial z} H^{(k)}(z, s) \\ = -k \left(\frac{1}{1-z} H^{(k-1)}(z, s) + z \frac{\partial}{\partial z} H^{(k-1)} \right)\end{aligned}$$

The coefficients on the left-hand side of this equation are the same as in (3.8), only the vector on the right-hand side changes. Hence, we can reuse $\Gamma_H(s)$ with substitution $s = 0$ from (3.9) and calculate the $\vec{h}_k$ vectors by updating only the right-hand side. $\square$

Note, that once $\Gamma_H(0)$ and M_1 are calculated the moments are obtained very efficiently.

3.3.3 Numerical results

Since the LST is given by $W(s) = (1 - \rho)G(\rho, s) = H(\rho, s)$, we choose a circle with radius ρ in the complex plane and solve $G(z, s)$ and $H(\rho, s)$ for n points around this circle denoted by the vector $\vec{g}$ and $\vec{h}$, respectively. Table 3.4 displays the parameters used for the test scenarios. For the light and medium loads, we use $n = 64$ equidistant points on the circle, and $n = 128$ points are used for the heavy load scenario. The number of points is determined experimentally by increasing n until further improvement in accuracy becomes negligible. Since the problem is

Table 3.4: The parameters for the tested scenarios during the numerical comparison

Scenario	$\rho = \lambda/\mu$	n
Light Load	1/3	64
Medium Load	2/3	64
Heavy Load	9/10	128

Table 3.5: The maximum errors calculated for the test scenarios at $s = 1$ and $s = 10 - 30i$.

Scenario	s	$\max err_G$	$\max err_H$	s	$\max err_G$	$\max err_H$
Light Load	1	0.0004	0.00006	10 - 30i	0.0002	0.0001
Medium Load	1	0.002	0.0005	10 - 30i	0.0015	0.001
Heavy Load	1	0.01	0.002	10 - 30i	0.005	0.004

one-dimensional, even using more than 100 points results in computation times of only a few seconds. We compare the results obtained by the approximation method to the exact solution — obtained in Section 3.2 — at three different stages: In the first stage, we assess the values obtained for $\vec{g}$ and $\vec{h}$; in the second stage, we evaluate the approximation of the LST; and in the final stage, we analyse the density obtained by inverting the LST.

Although the functional equations for the JSQ-PS model examined in Chapter 4 are more complex, the asymptotic accuracy and cost analysis of the approximation method remain the same and are therefore not repeated here, but rather deferred to Section 4.4.4.

Evaluation of $\vec{g}$ and $\vec{h}$

We calculate the estimates of $\vec{g}$ and $\vec{h}$ together with the exact values for two inputs: $s = 1$ and $s = 10 - 30i$ for each scenario. The maximum errors for each scenario are shown in Table 3.5. The errors are calculated using the simple formula

$$\max \left(\frac{\text{appr}(z_i) - \text{exact}(z_i)}{\text{exact}(z_i)} \mid 1 \leq i \leq n \right).$$

There are two observations to be made. Firstly, the accuracy decreases as the load increases. However, the estimates are extremely accurate at this stage, the highest maximum error being

Table 3.6: Comparing the estimates for the LST at $s = 1$ obtained by function G and H . The exact results are calculated by the no-conditioning method.

Scenario	$(1 - \rho)G(\rho, 1)$	$H(\rho, 1)$	Exact	err_G	err_H
Light Load	0.680501	0.680394	0.68041	0.0001	0.00002
Medium Load	0.554368	0.5527	0.553003	0.002	0.0005
Heavy Load	0.588025	0.579571	0.580986	0.006	0.002

Table 3.7: Comparing the estimates for the LST at $s = 10 - 30i$ obtained by function G and H . The exact results are calculated by the no-conditioning method.

Scenario	$(1 - \rho)G(\rho, 10 - 30i)$	$H(\rho, 10 - 30i)$	Exact	err_G	err_H
Light L.	$0.0294 + 0.0684i$	$0.0294 + 0.0684i$	$0.0294 + 0.0684i$	0.00006	0.00005
Medium L.	$0.0195 + 0.0467i$	$0.0195 + 0.0466i$	$0.0195 + 0.0466i$	0.001	0.0003
Heavy L.	$0.0344 + 0.0654i$	$0.0344 + 0.0651i$	$0.0343 + 0.0651i$	0.004	0.001

0.4% for the function H and 1% for the function G . Secondly, using the smoother function H outperforms G every single time.

Evaluation of the LSTs

Next, we move on to comparing the LSTs calculated by the previously obtained $\vec{g}$ and $\vec{h}$ values. Tables 3.6 and 3.7 show the generating functions evaluated at $z = \rho$ for parameter $s = 1$ calculating $W(1)$ and $s = 10 - 30i$ calculating $W(10 - 30i)$, respectively.

Consistent with the observation at the previous stage, the smoother function H outperforms the estimates produced by using the original function G . The errors either stayed the same or decreased compared to Table 3.5, which is of course expected as at the previous stage we displayed the maximum error of the n values estimated, whereas now — due to using a radius of $r = \rho$ throughout the calculations — we pick one element from the vectors compared before.

Evaluation of the calculated density functions

We use the Fixed Talbot inversion algorithm as seen in Section 3.2.1 to invert the Laplace transforms calculated by both the exact (no-conditioning) method and the approximation.

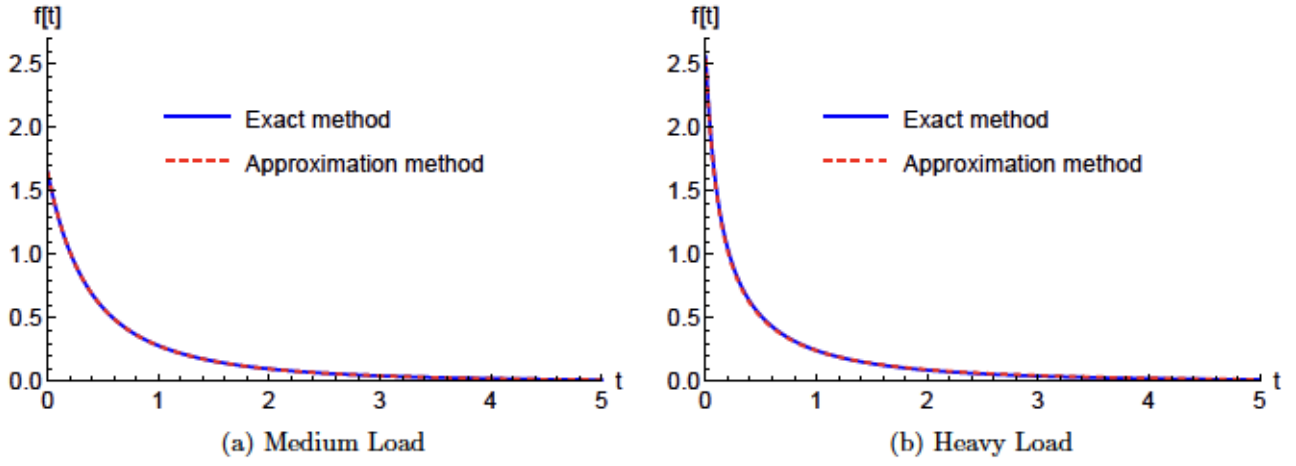


Figure 3.8: Response time density in an $M/M/1$ PS queue with parameters $\lambda = 2, \mu = 3$ (Medium Load) and $\lambda = 9, \mu = 10$ (Heavy Load).

Table 3.8: Response time density in an $M/M/1$ -PS queue at selected inputs under light load.

t	$f(t)$ appr	$f(t)$ exact	Diff
0.3	1.07383	1.07391	0.00009
0.6	0.526444	0.526485	0.00004
0.9	0.279384	0.279369	0.00002
1.2	0.157359	0.157322	0.00004
1.5	0.0927374	0.0926991	0.00004
1.8	0.0566157	0.0565842	0.00003
2.1	0.0355497	0.0355269	0.00002
2.4	0.02284	0.0228253	0.00001
2.7	0.0149568	0.0149486	0.00001
3.0	0.0099537	0.0099502	0.00000

Given that the smoother function H worked better in the previous steps, we proceed by inverting the LST expressed by H in the case of the approximation method. The obtained density function for the medium and high load scenarios is plotted in Figure 3.8, the density functions overlap perfectly. Therefore, for better comparison, we evaluate the response time density at selected inputs and calculate the difference. The results are displayed in Tables 3.8, 3.9 and 3.10 for each scenario.

Note that at this stage, the value of the density $f(t)$ is a numerical approximation of the function in both the no-conditioning and approximation methods, as it involves inverting the LSTs. The key difference lies in the accuracy of the inverted LSTs. In the case of the heavy load, the numbers match to order 10^{-3} and even to order 10^{-5} in the case of the light load.

Table 3.9: Response time density in an M/M/1-PS queue at selected inputs under medium load.

t	$f(t)$ appr	$f(t)$ exact	Diff
0.3	0.831743	0.832509	0.00077
0.6	0.491369	0.492029	0.00066
0.9	0.319285	0.319661	0.00038
1.2	0.220553	0.220691	0.00014
1.5	0.158914	0.158886	0.00003
1.8	0.118088	0.117957	0.00013
2.1	0.0898484	0.0896578	0.00019
2.4	0.0696551	0.0694361	0.00022
2.7	0.0548316	0.0546045	0.00023
3.0	0.0437158	0.0434935	0.00022

Table 3.10: Response time density in an M/M/1-PS queue at selected inputs under high load.

t	$f(t)$ appr	$f(t)$ exact	Diff
0.3	0.788716	0.790778	0.00206
0.6	0.429306	0.431811	0.0025
0.9	0.274764	0.276538	0.00177
1.2	0.190668	0.191727	0.00106
1.5	0.139097	0.139612	0.00052
1.8	0.105045	0.105178	0.00013
2.1	0.0813854	0.0812611	0.00012
2.4	0.064319	0.0640286	0.00029
2.7	0.0516473	0.0512557	0.00039
3.0	0.0420191	0.0415719	0.00045

Calculating the first 4 moments

Finally, we compare the first four moments calculated by the approximation method and the exact method. In the case of the approximation method, we use Lemma 3.3.1 to calculate the moments recursively. Again, since the circle the discretisation points lie on has radius ρ , the k th moment is at the index of the vector corresponding to input ρ , therefore no interpolation is required once h_k is obtained. In order to calculate the moments from the exact LST, Coffman's method has been chosen as it is more convenient to calculate the k th derivative with respect to s numerically in that case. The results are shown in Tables 3.11, 3.12 and 3.13.

Table 3.11: First four moments of the response time density in an M/M/1-PS queue under light load.

Moment	Approximation method	Exact method	Diff(%)
1st	0.5000	0.5000	0.000
2nd	0.6002	0.6000	0.032
3rd	1.2606	1.2600	0.051
4th	3.9909	3.9909	0.001

Table 3.12: First four moments of the response time density in an M/M/1-PS queue under medium load.

Moment	Approximation method	Exact method	Diff(%)
1st	1.0000	1.0000	0.000
2nd	3.0134	3.0000	0.445
3rd	18.118	18.000	0.654
4th	176.16	176.40	0.136

Table 3.13: First four moments of the response time density in an M/M/1-PS queue under high load.

Moment	Approximation method	Exact method	Diff(%)
1st	1.0000	1.000	0.000
2nd	3.6812	3.6363	1.234
3rd	29.326	28.760	1.966
4th	393.62	392.58	0.272

The calculated approximate moments are extremely accurate in the cases of light and medium loads. Even under heavy load, the difference remains below 2%. However, it is important to note that in this scenario the utilisation is 90% which is unlikely to occur in real-world systems. This particular scenario is deliberately chosen to test the limits of the approximation method. To achieve the displayed accuracy, we increase the number of points from 128 to 200 when computing the moments under heavy load. Despite this increase, the computation still only takes a few seconds. Overall, we conclude that the approximation method performs exceptionally well, and the results obtained so far are highly promising.

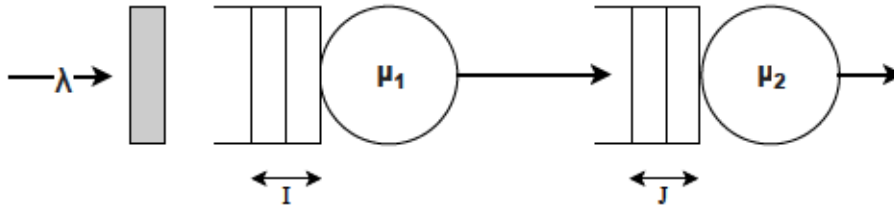


Figure 3.9: Two PS queues in tandem with arrival rate λ and service rates μ_1, μ_2 .

3.4 Tandem PS queues

In this section, we calculate the response time density for a tandem pair of PS queues, for the first time, to our knowledge. As detailed in Chapter 2, the functional equations for the LST of the response time distribution of two PS queues in tandem have been obtained in [CJFM86]. However, attempts at finding an explicit solution to these equations have been unsuccessful and instead the problem is reduced to a boundary value problem. We derive similar equations, for the generating functions we get by applying the transformation used for the M/M/1-PS case. We then solve these equations using the approximation method introduced in the previous section. Furthermore, a recursive formula for the moments of the response time distribution is derived. The results are evaluated by comparing them to regenerative simulation and show high accuracy.

3.4.1 Obtaining the functional equations

We start by obtaining the functional equations for the generating function of the LST of conditional response time densities. Let λ denote the arrival rate and μ_1, μ_2 the service rates at queues 1 and 2, respectively. The queueing system is illustrated in Figure 3.9. It is known that the stationary queue-length distribution has a product form, and it is given by the formula

$$\pi_{ij} = (1 - \rho_1)(1 - \rho_2)\rho_1^i\rho_2^j$$

where $\rho_1 = \lambda/\mu_1, \rho_2 = \lambda/\mu_2$, and $\lambda < \min(\mu_1, \mu_2)$ is required for the stationary distribution to exist. Let $\Phi_1(i, j, s)$ denote the LST of the remaining conditional response time distribution of

a task at queue 1 when the queue lengths are $I = i$ and $J = j$ at queues 1 and 2, and define the corresponding generating function as

$$G(x, y, s) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \Phi_1(i, j, s) x^i y^j$$

Analogously for a task at queue 2, we denote the conditional LSTs by $\Phi_2(i, j, s)$ and the corresponding generating function by

$$F(x, y, s) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \Phi_2(i, j, s) x^i y^j$$

Due to the product form stationary queue-length probabilities, the LST of the response time distribution is straightforwardly calculated as $W(s) = (1 - \rho_1)(1 - \rho_2)G(\rho_1, \rho_2, s)$.

The functional equations for G and F in [CJFM86] are derived using the same approach as in the M/M/1-PS case discussed in Section 3.2. In Section 3.3, we have shown that a simple transformation of the generating functions enhances the accuracy of the numerical approximation. Here, we have an extra variable in the generating functions, however, when $s = 0$, the LST of the remaining sojourn time of a task at queue 1, $\Phi_1(i, j, s)$, simplifies to the integral of the density function and therefore $\Phi_1(i, j, 0) = 1$. Furthermore, for $\Re(s) > 0$, $|\Phi_1(i, j, s)| < 1$, for all $i, j \geq 0$ as

$$|\Phi_1(i, j, s)| \leq \int_0^{\infty} f(t) |e^{-st}| dt = \int_0^{\infty} f(t) e^{-\Re(s)t} dt < 1$$

Of course, the same applies to $\Phi_2(i, j, s)$. Thus, $G(x, y, 0) = F(x, y, 0) = \frac{1}{(1-x)(1-y)}$ and for $\Re(s) > 0$, $|G(x, y, s)| < \frac{1}{(1-x)(1-y)}$. We define $H(x, y, s) = (1-x)(1-y)G(x, y, s)$ and $J(x, y, s) = (1-x)(1-y)F(x, y, s)$. Then we have $H(x, y, 0) = J(x, y, 0) = 1$. These constant functions are, again, numerically more desirable than $G(x, y, s)$ and $F(x, y, s)$ with their sharp peaks near $x = 1$ and $y = 1$. The LST is now given by $W(s) = H(\rho_1, \rho_2, s)$. The following proposition states the functional equations for H and J . It is presented with a sketch proof as the details of the omitted steps can be found in [CJFM86].

Proposition 3.4.1. *The generating functions $H(x, y, s)$ and $J(x, y, s)$ are given by the following equations.*

$$\begin{aligned}
H(x, y, s) & \left(\mu_2 y + \frac{sy - \mu_1 x + \mu_1 y}{(1 - y)} \right) \\
& - \frac{\partial H(x, y, s)}{\partial x} \left(\frac{(1 - x)[x(x - y)\mu_1 + y(\lambda - x(s + \lambda) - x(1 - y)\mu_2)]}{1 - y} \right) \\
& + H(x, 0, s)(\mu_1 x - \mu_2 y) + \frac{\partial H(x, 0, s)}{\partial x}(1 - x)x(\mu_1 x - \mu_2 y) = \frac{\mu_1(1 - x)y}{(1 - y)} J(x, y, s) \quad (3.10)
\end{aligned}$$

$$\begin{aligned}
J(x, y, s) & \left(\frac{\lambda(1 - 1/x) + s + \mu_1(1 - x)}{1 - y} + \mu_2 \right) \\
& + \frac{\partial}{\partial y} J(x, y, s) (y[\lambda(1 - 1/x) + \mu_2(1 - y) + s] + \mu_1(y - x)) \\
& + J(0, y, s) \frac{1 - x}{1 - y} \left(\frac{\lambda}{x} - \mu_1 \right) + \frac{\partial}{\partial y} J(0, y, s)(1 - x)y \left(\frac{\lambda}{x} - \mu_1 \right) = \mu_2 \quad (3.11)
\end{aligned}$$

Proof. We start with a task at queue 2. A recurrence formula is derived for the density of conditional remaining response time at time t by expressing it as the sum of h — an infinitesimal initial interval — and the remaining conditional response time at time $t + h$. The resulting equation in terms of the conditional LSTs is [CJFM86][Equation (2.3)]

$$\begin{aligned}
\Phi_2(i, j, s) (\lambda + \mu_1 I_{i>0} + \mu_2 + s) & = \lambda \Phi_2(i + 1, j, s) \\
& + \mu_1 I_{i>0} \Phi_2(i - 1, j + 1, s) + \mu_2 \frac{j}{j + 1} \Phi_2(i, j - 1, s) + \frac{\mu_2}{j + 1}
\end{aligned}$$

Multiplying by $(j + 1)x^i y^j$ and summing for all i and j values, after repeated but routine transformations of the resulting double sums, we get

$$\begin{aligned}
F(x, y, s) & \left(\lambda \left(1 - \frac{1}{x} \right) + \mu_1 + \mu_2(1 - y) + s \right) \\
& + \frac{\partial}{\partial y} F(x, y, s) \left(y \left(\lambda \left(1 - \frac{1}{x} \right) + \mu_2(1 - y) + s \right) + \mu_1(y - x) \right) \\
& + F(0, y, s) \left(\frac{\lambda}{x} - \mu_1 \right) + \frac{\partial}{\partial y} F(0, y, s) \left(y \left(\frac{\lambda}{x} - \mu_1 \right) \right) = \frac{\mu_2}{(1 - x)(1 - y)}
\end{aligned}$$

Multiplying both sides by $(1-x)(1-y)$ and using the definition of $J(x, y, s)$ to replace the F terms, after some rearrangement we have

$$\begin{aligned} J(x, y, s) & \left(\frac{\lambda(1-1/x) + s + \mu_1(1-x)}{1-y} + \mu_2 \right) \\ & + \frac{\partial}{\partial y} J(x, y, s) (y[\lambda(1-1/x) + \mu_2(1-y) + s] + \mu_1(y-x)) \\ & + J(0, y, s) \frac{1-x}{1-y} \left(\frac{\lambda}{x} - \mu_1 \right) + \frac{\partial}{\partial y} J(0, y, s) (1-x)y \left(\frac{\lambda}{x} - \mu_1 \right) = \mu_2 \end{aligned}$$

Moving on to a task at queue 1, by the same reasoning, the recurrence formula, in this case, is [MM83][Equation (2.5)]

$$\begin{aligned} \Phi_1(i, j, s) (\lambda + \mu_1 + \mu_2 I_{j>0} + s) & = \lambda \Phi_1(i+1, j, s) \\ & + \mu_1 \left(\frac{i}{i+1} \Phi_1(i-1, j+1, s) + \frac{1}{i+1} \Phi_2(i, j, s) \right) + \mu_2 \Phi_1(i, j-1, s) \end{aligned}$$

Multiplying by $(i+1)x^i y^j$ and summing for all i and j values, the resulting equation is

$$\begin{aligned} G(x, y, s) & \left(\lambda + \mu_1 \left(1 - \frac{x}{y} \right) + \mu_2(1-y) + s \right) \\ & + \frac{\partial}{\partial x} G(x, y, s) \left(\lambda(x-1) + \mu_1 x \left(1 - \frac{x}{y} \right) + \mu_2 x(1-y) + xs \right) \\ & + G(x, 0, s) \left(\mu_1 \frac{x}{y} - \mu_2 \right) + \frac{\partial}{\partial x} G(x, 0, s) \left(x \left(\mu_1 \frac{x}{y} - \mu_2 \right) \right) = \mu_1 F(x, y, s) \end{aligned}$$

Multiplying by $(1-x)^2 y$ and using the definitions of $H(x, y, s)$ and $J(x, y, s)$ to replace the G and F terms, respectively, after some transformation, we have

$$\begin{aligned} H(x, y, s) & \left(\mu_2 y + \frac{sy - \mu_1 x + \mu_1 y}{(1-y)} \right) \\ & - \frac{\partial}{\partial x} H(x, y, s) \left(\frac{(1-x)[x(x-y)\mu_1 + y(\lambda - x(s+\lambda) - x(1-y)\mu_2)]}{1-y} \right) \\ & + H(x, 0, s) (\mu_1 x - \mu_2 y) + \frac{\partial}{\partial x} H(x, 0, s) (1-x)x (\mu_1 x - \mu_2 y) = \mu_1 \frac{y(1-x)}{1-y} J(x, y, s) \end{aligned}$$

□

Note, that while (3.11) has only J terms, (3.10) has both H and J terms. Therefore, a solution must first be obtained for (3.11) which is then plugged into (3.10), after which it can be solved.

3.4.2 Solving the functional equations

In the previous chapter, once s is treated as a fixed parameter, the functional equation is solved in one variable. In this case, however, we have two queues, and therefore, treating s as a parameter once more, the generating functions and resulting functional equations have two variables. Therefore, we first briefly explain how the approximation method is extended to handle two dimensions.

Approximation method in 2 dimensions

Consider a bivariate functional equation of the form

$$a(x, y)G(x, y) = b(x, y) + c(x, y)G(x, 0) + d(x, y)\frac{\partial}{\partial y}G(0, y) \quad (3.12)$$

where $a(x, y)$, $b(x, y)$, $c(x, y)$ and $d(x, y)$ are known functions of x and y . Let C_x and C_y be contours in the complex planes of x and y . Typically, C_x and C_y are circles. The interior of C_x is then a disk D_x , and similarly for C_y, D_y . Now, the value of a function $f(x, y)$ at any point (x_0, y_0) inside the Cartesian product of the contours, $C_x \times C_y$ (i.e. with $x_0 \in D_x$, and $y_0 \in D_y$), can be obtained by Cauchy's Integral Theorem applied twice in two dimensions, provided $f(x, y)$ is known for all $x \in C_x, y \in C_y$; see [Hor73, Pri03] for example. We therefore first seek an interpolation for f over $\{(x, y) \in C_x \times C_y\}$ by choosing two sets of discretisation points $x_1, \dots, x_m \in C_x$ and $y_1, \dots, y_n \in C_y$. We take $m = n$ to simplify the notation; the analysis is no more complicated when $m \neq n$ but is more cluttered. An approximation to $f(x, b)$ on C_x , for fixed $b \in C_y$, is an interpolation over the points $x_1, \dots, x_n$ of the values $f(x_1, b), \dots, f(x_n, b)$, and similarly for $f(a, y)$ at fixed $a \in C_x$. We order the point-pairs lexicographically by subscript, $\{(x_1, y_1), \dots, (x_1, y_n), (x_2, y_1), \dots, (x_n, y_n)\}$ and say that $\vec{f} = (f((x_1, y_1), \dots, f(x_1, y_n), f(x_2, y_1), \dots, f(x_n, y_n)))$ is a *vector representation* of f on $C_x \times C_y$. The vector representation can even be symbolic, i.e. $\vec{f} = (f_{11}, \dots, f_{nn})$.

Table 3.14: Matrices representing operations in two dimensions.

Operation	Matrix
Identity operator giving f	$I^{nm \times nm}$
Multiplication by known function $A(x, y)$	$\Lambda(A)$
Differentiation with respect to x	$M_{1,0}$
Differentiation w.r.t. y	$M_{0,1}$
Evaluation at an internal point $u_0 \in D_x$	U_{u_0}
Evaluation at an internal point $v_0 \in D_y$	V_{v_0}

To transform the functional equation into a matrix-vector equation, all we need is the *matrix representation* of each operator in the functional equation. For example, analogous to the one-dimensional case, multiplication by a fixed function $a(x, y)$ is represented by a diagonal matrix of the values of the function at the discretisation point-pairs, ordered lexicographically, and we define

$$\Lambda(\vec{a}) = \text{Diag}\left(a(x_i, y_j) \mid 1 \leq i \leq n, 1 \leq j \leq n\right)$$

Matrices for the operators of partial differentiation and evaluation at an internal point have been obtained in [Har24]. The details are not reproduced here, apart from Table 3.14 which lists the matrices corresponding to each operator of interest and is included here to enhance readability. Note that all the matrices in Table 3.14 are constant, applying to any functional equation solved for the given n discretization points in both complex planes. Equation (3.12) above is thereby transformed to

$$(\Lambda(\vec{a}) - \Lambda(\vec{c})V_0 - \Lambda(\vec{d})M_{0,1}U_0)\vec{g} = \Lambda(\vec{b})$$

Calculating the LST

The functional equation (3.11) of $J(x, y, s)$ at the chosen discretisation point-pairs, represented by the lexicographically ordered vector $\vec{j} = (J(x_i, y_j) \mid 1 \leq i \leq m, 1 \leq j \leq n)$, is transformed

into the matrix-vector equation

$$\left(\Lambda(\vec{a}_1) + \Lambda(\vec{a}_2)M_{0,1} + \Lambda(\vec{a}_3)U_0 + \Lambda(\vec{a}_4)M_{0,1}U_0 \right) \vec{j} = (\mu_2, \dots, \mu_2)^\top$$

where $\vec{a}_1, \vec{a}_2, \vec{a}_3, \vec{a}_4$ are the vector representations of the functions A_1, A_2, A_3, A_4 defined by

$$\begin{aligned} A_1(x, y) &= \left(\frac{\lambda(1 - 1/x) + s + \mu_1(1 - x)}{1 - y} + \mu_2 \right) \\ A_2(x, y) &= (y[\lambda(1 - 1/x) + \mu_2(1 - y) + s] + \mu_1(y - x)) \\ A_3(x, y) &= \frac{1 - x}{1 - y} \left(\frac{\lambda}{x} - \mu_1 \right) \\ A_4(x, y) &= (1 - x)y \left(\frac{\lambda}{x} - \mu_1 \right) \end{aligned}$$

Define $\Gamma_J(s)$ as

$$\Gamma_J(s) = \left(\Lambda(\vec{a}_1) + \Lambda(\vec{a}_2)M_{0,1} + \Lambda(\vec{a}_3)U_0 + \Lambda(\vec{a}_4)M_{0,1}U_0 \right)^{-1} \quad (3.13)$$

Note that as the notation suggests, $\Gamma_J(s)$ depends on the value of the Laplace parameter s .

The solution for $\vec{j}$ for a given s value can be calculated as

$$\vec{j} = \Gamma_J(s)(\mu_2, \dots, \mu_2)^\top \quad (3.14)$$

Analogously, (3.10) is transformed into the matrix-vector equation

$$\left(\Lambda(\vec{b}_1) - \Lambda(\vec{b}_2)M_{1,0} + \Lambda(\vec{b}_3)V_0 + \Lambda(\vec{b}_4)M_{1,0}V_0 \right) \vec{h} = \Lambda(\vec{b}_5)\vec{j}$$

where $\vec{b}_i, 1 \leq i \leq 5$ are the vector representations of the functions B_i defined by

$$\begin{aligned} B_1(x, y) &= \mu_2 y + \frac{sy - \mu_1 x + \mu_1 y}{(1 - y)} \\ B_2(x, y) &= \frac{(1 - x)[x(x - y)\mu_1 + y(\lambda - x(s + \lambda) - x(1 - y)\mu_2)]}{1 - y} \\ B_3(x, y) &= \mu_1 x - \mu_2 y \\ B_4(x, y) &= (1 - x)x(\mu_1 x - \mu_2 y) \\ B_5(x, y) &= \frac{\mu_1(1 - x)y}{(1 - y)} \end{aligned}$$

Define $\Gamma_H(s)$ as

$$\Gamma_H(s) = \left(\Lambda(\vec{b}_1) - \Lambda(\vec{b}_2)M_{1,0} + \Lambda(\vec{b}_3)V_0 + \Lambda(\vec{b}_4)M_{1,0}V_0 \right)^{-1} \quad (3.15)$$

Again, $\Gamma_H(s)$ depends on the value of the parameter s . The solution for $\vec{h}$ for a given s value can be calculated as

$$\vec{h} = \Gamma_H(s)\Lambda(\vec{b}_5)\vec{j}$$

where $\vec{j}$ is given by (3.14).

Calculating the moments

Moments are computed by the method used in the M/M/1-PS case in Section 3.3.2. That is, let m_n denote the n -th moment, then

$$m_n = (-1)^n \frac{\partial^n}{\partial s^n} W(0) = (-1)^n \frac{\partial^n}{\partial s^n} H(\rho_1, \rho_2, 0)$$

The following lemmas, analogous to Lemma 3.3.1, derive a recursive formula for obtaining the derivatives of $J(x, y, s)$ and $H(x, y, s)$ with respect to s using their vector representation $\vec{j}$ and $\vec{h}$, respectively.

Lemma 3.4.1. *The k th partial derivative of $J(x, y, s)$ with respect to s at $s = 0$ has vector representation $\vec{j}_k$ given by the recurrence*

$$\begin{aligned} \vec{j}_0 &= (1, \dots, 1)^\top \\ \vec{j}_k &= \Gamma_J(0) \left(-k\Lambda(\vec{c}_1)\vec{j}_{k-1} - k\Lambda(\vec{c}_2)j'_{k-1} \right) \\ j'_k &= M_{0,1}\vec{j}_k \end{aligned}$$

where $\Gamma_J(0)$ is given by (3.13) with substitution $s = 0$ and $\vec{c}_1, \vec{c}_2$ are the vector representations of functions C_1, C_2 defined as

$$C_1(x, y) = \frac{1}{1-y}$$

$$C_2(x, y) = y$$

Proof. First, note that $J(x, y, 0) = 1$ by construction. Differentiating (3.11) k times with respect to s , we get

$$\begin{aligned}
J^{(k)}(x, y, s) & \left(\frac{\lambda(1 - 1/x) + s + \mu_1(1 - x)}{1 - y} + \mu_2 \right) \\
& + \frac{\partial}{\partial y} J^{(k)}(x, y, s) (y[\lambda(1 - 1/x) + \mu_2(1 - y) + s] + \mu_1(y - x)) \\
& + J^{(k)}(0, y, s) \frac{1 - x}{1 - y} \left(\frac{\lambda}{x} - \mu_1 \right) + \frac{\partial}{\partial y} J^{(k)}(0, y, s) (1 - x)y \left(\frac{\lambda}{x} - \mu_1 \right) \\
& = -k \left(\frac{1}{1 - y} J^{(k-1)}(x, y, s) + y \frac{\partial}{\partial y} J^{(k-1)}(x, y, s) \right)
\end{aligned}$$

The coefficients on the left-hand side of this equation are the same as in (3.11), only the vector on the right-hand side changes. Hence, we can reuse $\Gamma_J(s)$ with substitution $s = 0$ from (3.13) and calculate the $\vec{j}_k$ vectors very efficiently by updating only the right-hand side. $\square$

Once the $\vec{j}_k$ vectors are calculated, the derivative of $H(x, y, s)$ with respect to s can be calculated using the following lemma.

Lemma 3.4.2. *The k th partial derivative of $H(x, y, s)$ with respect to s at $s = 0$ has vector representation $\vec{h}_k$ given by the recurrence*

$$\begin{aligned}
\vec{h}_0 &= (1, \dots, 1)^\top \\
\vec{h}_k &= \Gamma_H(0) \left(-k\Lambda(\vec{d}_1)\vec{h}_{k-1} - k\Lambda(\vec{d}_2)h'_{k-1} + \Lambda(\vec{d}_3)\vec{j}_k \right) \\
h'_k &= M_{1,0}\vec{h}_k
\end{aligned}$$

where $\Gamma_H(0)$ is given by (3.15) with substitution $s = 0$, $\vec{j}_k$ is given by Lemma 3.4.1 and $\vec{d}_1, \vec{d}_2, \vec{d}_3$ are vector representations of functions D_1, D_2, D_3 defined as

$$\begin{aligned}
D_1(x, y) &= \frac{y}{1 - y} \\
D_2(x, y) &= \frac{(1 - x)xy}{1 - y} \\
D_3(x, y) &= \mu_1 \frac{y(1 - x)}{1 - y}
\end{aligned}$$

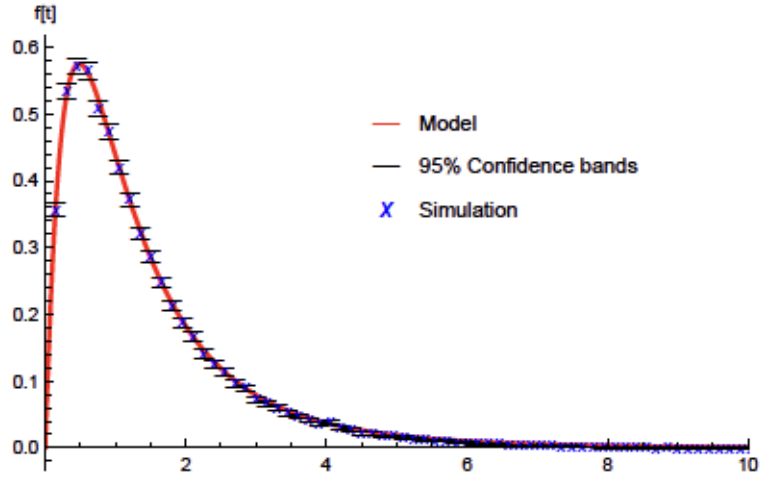


Figure 3.10: Response time density for a two-server PS tandem network with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$.

Proof. Again, we start by noting that $H(x, y, 0) = 1$ by construction. Calculating the k th derivative of (3.10) with respect to s , we have

$$\begin{aligned}
 H^{(k)}(x, y) & \left(\mu_2 y + \frac{sy - \mu_1 x + \mu_1 y}{(1 - y)} \right) \\
 & - \frac{\partial H^{(k)}(x, y)}{\partial x} \left(\frac{(1 - x)[x(x - y)\mu_1 + y(\lambda - x(s + \lambda) - x(1 - y)\mu_2)]}{1 - y} \right) \\
 & + H^{(k)}(x, 0) (\mu_1 x - \mu_2 y) + \frac{\partial H^{(k)}(x, 0)}{\partial x} (1 - x)x (\mu_1 x - \mu_2 y) + kH^{(k-1)}(x, y) \left(\frac{y}{1 - y} \right) \\
 & + k \frac{\partial H^{(k-1)}(x, y)}{\partial x} \left(\frac{(1 - x)xy}{1 - y} \right) = \frac{\mu_1(1 - x)y}{(1 - y)} J^{(k)}(x, y, s)
 \end{aligned}$$

Once more, the coefficients on the left-hand side are the same as in (3.10), only the vector on the right-hand side changes. Hence, we can reuse $\Gamma_H(s)$ with substitution $s = 0$ from (3.15) and calculate the $\vec{h}_k$ vectors very efficiently by updating only the right-hand side. $\square$

3.4.3 Numerical results

As the generating function $H(x, y, s)$ needs to be evaluated at $x = \rho_1, y = \rho_2$ to express the LST of the response time distribution, we choose C_x, C_y to be circles centred at the origin with radii $r_1 = \rho_1, r_2 = \rho_2$ in the complex plane. Therefore, $H(\rho_1, \rho_1, s)$ is calculated from its vector representation $\vec{h}$ by selecting the index corresponding to the discretisation point pair (ρ_1, ρ_2) , or if the point falls within two discretisation points, by linear interpolation. We use

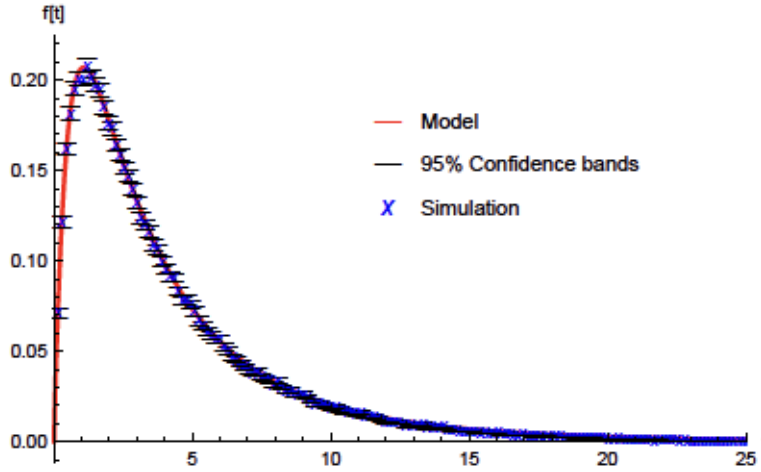


Figure 3.11: Response time density for a two-server PS tandem network with parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$.

$n = 64$ points in both dimensions, determined through experimentation, as the improvement in accuracy becomes negligible when n is increased further. However, since we are working in two dimensions, the matrices have size $n^2 \times n^2$, and the computational cost grows rapidly as a result, with a detailed analysis presented in Section 4.4.4.

We use regenerative simulation to compare the results, following the approach described in [Gly06]. Specifically, we start with both queues being empty and simulate arrivals and departures; each time both queues become empty again, we mark the end of one regeneration cycle. The total simulation runs for 5×10^5 regeneration cycles. We estimate the empirical density function using data from the entire simulation. Following [Chi95], the sets of regeneration cycles are partitioned into batches, the sizes of which are chosen equal to the square root of the number of cycles, or nearest integer; here 500, giving a total of 500 batches as well. For each batch, we compute the response time moments and then average these across all batches to obtain the final empirical moments. The same batching approach is also used to construct the 95% confidence intervals for the empirical density estimates.

We run the simulation for three sets of parameters. The first two sets are chosen in accordance with [Har24], allowing us to compare our results with a PS-FCFS tandem, that is, an $M/M/1$ -PS queue followed by an $M/M/1$ -FCFS queue, using the same parameters. These are: a light load case with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$, resulting in a 50% utilisation at the first queue and $\approx 34\%$ at the second queue, and a heavy load case with parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$

Table 3.15: Response time moments for two PS queues in tandem with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$.

Moment	PS-PS model	Simulation	95% CB
1st	1.50000	1.50137	± 0.00542
2nd	4.36580	4.38277	± 0.04519
3rd	21.58840	21.78040	± 0.50304
4th	163.547	165.567	± 7.65520

giving utilisations $\approx 71\%$ and $\approx 67\%$ at queues 1 and 2, respectively. In addition, we also run a very heavy load case with parameters $\lambda = 2$, $\mu_1 = 2.4$, $\mu_2 = 2.6$ resulting in utilisations 83.34% and 77% at the two queues. This stresses the method, since higher utilisation leads to less smooth generating functions that are harder to approximate. In all cases, we use the Fixed Talbot method to invert the LST as described in the M/M/1-PS case in Section 3.2.1.

Figures 3.10, 3.11, and 3.12 show the obtained density together with the simulation-generated relative frequency histogram with 95% confidence bands. The results are remarkably accurate with most of the model points falling within the confidence intervals, even in the very heavy load case with utilisations 83.34% and 77% at the two queues.

Next, we calculate the moments using the recursive method given by Lemma 3.4.2. Note that this is much faster than calculating the density functions, since for the moments we only need to calculate $\Gamma_H(s)$ and $\Gamma_J(s)$ at $s = 0$ to obtain all four moments for a given set of parameters, whereas the density function requires the LST to be evaluated at 32 different s -values to perform the inversion at a single point. Tables 3.15, 3.16 and 3.17 display the first four moments calculated by our model, together with the moments calculated from simulation with the corresponding 95% confidence bands. In the cases of heavy load and very heavy load, we increase the number of discretisation points to $n = 120$ to calculate the moments as the results obtained with $n = 64$ are less accurate. All the moments calculated by the model fall within the confidence intervals, except the fourth moment in Table 3.17, which is slightly outside the 95% confidence interval. However, we know that the first moments must be 1.5 and 4.5 and 4.16667 for the three cases, respectively, since these are equal to the first moment in an FCFS-FCFS tandem network, that is two M/M/1-FCFS queues placed after each other. The first moment

Table 3.16: Response time moments for two PS queues in tandem with parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$

Moment	PS-PS model	Simulation	95% CB
1st	4.50000	4.49004	± 0.01689
2nd	42.98960	42.77410	± 0.43598
3rd	725.31300	720.12600	± 15.18100
4th	18991.800	18821.00000	± 728.230

calculated by the model matches these numbers exactly, whereas the simulation is slightly off. Therefore, we conclude that the higher moments calculated by the model are likely to be more accurate than those of the simulation.

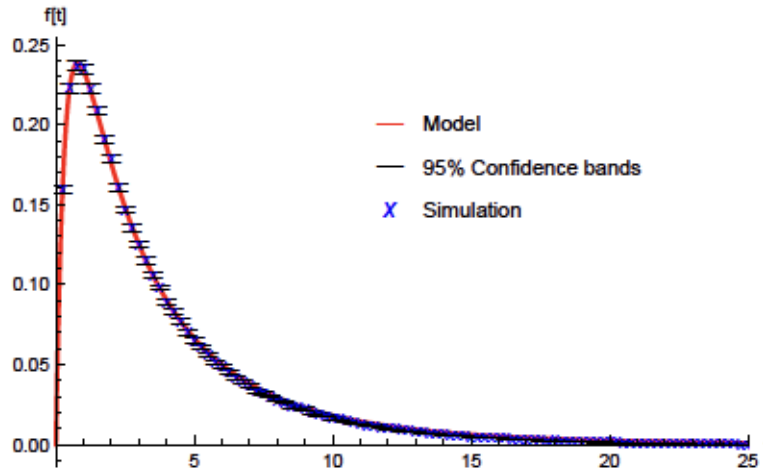


Figure 3.12: Response time density for a two-server PS tandem network with parameters $\lambda = 2$, $\mu_1 = 2.4$, $\mu_2 = 2.6$.

Experiments on the queue order

It is interesting to study the effect of reversing the order of the queues in the tandem on the response time. Tables 3.18, 3.19, and 3.20 show the first four moments of the original tandem queues for the light, heavy, and very heavy load cases (second column), as well as their order-reversed versions (third column). The moments of the original and reversed tandems are very close to each other, so we can conclude that, numerically, the order of the queues does not matter. Furthermore, these results suggest that the response times at the two queues may, in fact, be independent. To investigate this, we compute the response time moments of the

Table 3.17: Response time moments for two PS queues in tandem with parameters $\lambda = 2$, $\mu_1 = 2.4$, $\mu_2 = 2.6$

Moment	PS-PS model	Simulation	95% CB
1st	4.16667	4.16240	± 0.01833
2nd	39.75420	39.85860	± 0.47209
3rd	697.2140	711.71200	± 17.62730
4th	19311.60	20559.80000	± 992.76300

Table 3.18: Comparing response time moments for a PS tandem with rates $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$; its order-reversed version with $\mu_1 = 3$, $\mu_2 = 2$; and the convolution of two independent PS queues with $\lambda_1 = 1$, $\mu_1 = 2$ and $\lambda_2 = 1$, $\mu_2 = 3$.

Moment	Normal Order	Reversed Order	Convolution
1st	1.50000	1.50000	1.50000
2nd	4.36580	4.36586	4.26667
3rd	21.58840	21.58900	20.39300
4th	163.547	163.562	150.185

convolution of two independent PS queues with parameters λ, μ_1 and λ, μ_2 , and present the results in the last column of the tables. Note that the moments of the convolution are exact, as the moments of the individual PS queues can be calculated independently, reusing the methods used in Section 3.3.3. The moments of the convolutions differ significantly from their respective tandem counterparts. Therefore, we conclude that the response times at the two queues are not independent, but conjecture that the overall response time of the system is independent of the order of the queues.

Comparison to PS-FCFS and FCFS-FCFS tandems

Finally, we compare the first four moments of the PS-PS tandem model with those of a PS-FCFS tandem, that is, an $M/M/1$ -PS queue followed by an $M/M/1$ -FCFS queue, and an FCFS-FCFS tandem, that is, two $M/M/1$ -FCFS queues in series, using the same parameters. The numbers for the PS-FCFS and FCFS-FCFS models are taken from [Har24]. The results are given in Tables 3.21 and 3.22. We know from Little's theorem that the first moments across the three tandem networks must coincide, which is consistent with the numerical results. Furthermore, as expected, all higher moments increase when going from an FCFS to a PS schedule, with

Table 3.19: Comparing response time moments for a PS tandem with rates $\lambda = 1, \mu_1 = 1.4, \mu_2 = 1.5$; its order-reversed version with $\mu_1 = 1.5, \mu_2 = 1.4$; and the convolution of two independent PS queues with $\lambda_1 = 1, \mu_1 = 1.4$ and $\lambda_2 = 1, \mu_2 = 1.5$.

Moment	Normal Order	Reversed Order	Convolution
1st	4.50000	4.50000	4.50000
2nd	42.98960	42.98950	41.44440
3rd	725.31300	725.32900	658.53640
4th	18991.80000	18994.00000	16171.557

Table 3.20: Comparing response time moments for a PS tandem with rates $\lambda = 2, \mu_1 = 2.4, \mu_2 = 2.6$; its order-reversed version with $\mu_1 = 2.6, \mu_2 = 2.4$; and the convolution of two independent PS queues with $\lambda_1 = 2, \mu_1 = 2.4$ and $\lambda_2 = 2, \mu_2 = 2.6$.

Moment	Normal Order	Reversed Order	Convolution
1st	4.16667	4.16667	4.16667
2nd	39.75420	39.75240	38.78973
3rd	697.21400	697.17900	666.72056
4th	19311.60000	19314.90000	18749.68100

PS-PS producing the highest numbers, so that users experience greater variability. Recall, however, that the increased variability of PS scheduling is compensated by the fairness of the schedule, as the mean slowdown of a task in a PS queue is independent of its service requirement [BHB01], that is, the expected time a task spends in the system is proportional to its own service requirement [AF83]. As a result, smaller tasks do not get stuck behind large tasks as in a FCFS queue. This inherent fairness is why PS tends to be favoured over FCFS.

3.5 Summary

In this chapter, we calculate the response time density and corresponding moments for an M/M/1-PS queue and a PS-PS tandem network. The following points summarise the main contributions of the chapter:

- We obtain a novel, simplified functional equation for the generating function of the LST of response time distribution in an M/M/1-PS queue, without conditioning on the size of the tagged task. The equation is explicitly solved, and the resulting LST is efficiently inverted using the Fixed Talbot method.

Table 3.21: Comparing response time moments in PS-PS, PS-FCFS and FCFS-FCFS tandems with parameters $\lambda = 1$, $\mu_1 = 2$, $\mu_2 = 3$.

Moment	PS-PS model	PS-FCFS	FCFS-FCFS
1st	1.50000	1.49999	1.50
2nd	4.36580	4.25421	3.50
3rd	21.58840	20.5703	11.25
4th	163.547	149.59000	46.50

Table 3.22: Comparing response time moments in PS-PS, PS-FCFS and FCFS-FCFS tandems with parameters $\lambda = 1$, $\mu_1 = 1.4$, $\mu_2 = 1.5$

Moment	PS-PS model	PS-FCFS	FCFS-FCFS
1st	4.50000	4.49954	4.50
2nd	42.75710	38.53280	30.50
3rd	708.37300	574.1810	276.75
4th	17863.300	13833.000	3151.50

- We apply an approximation method to solve the same functional equation and conduct extensive validation by comparing the obtained results to the exact solution. Additionally, we derive a recursive formula for efficiently computing response time moments. Our findings confirm that the approximation method performs exceptionally well in calculating both the density function and the moments.
- We obtain the functional equations for the transformed generating functions of the LST of conditional response time distribution in a PS-PS tandem network. We apply the approximation method to solve the resulting system of functional equations. To the best of our knowledge, this is the first time the response time density has been obtained for this model. Additionally, we derive a recursive formula for computing the response time moments efficiently. Numerical results demonstrate that the density function and the first four moments calculated by the model exhibit very high accuracy when compared to regenerative simulation.

Chapter 4

Response time in a pair of PS queues with JSQ scheduling

4.1 Introduction

In this chapter, we derive the response time density in a pair of PS queues with JSQ scheduling for the first time. We consider heterogeneous service rates and arbitrary routing probabilities as outlined in Section 2.4.2, solving the most general form of the JSQ-PS model with two parallel queues. Although the approach follows the methodology used for the tandem PS-PS queue, this model is considerably more complex. One reason for this complexity is that the steady-state queue-length distribution no longer has a product form since arriving tasks join the shorter of the two queues; consequently, the queue lengths at queues 1 and 2 become inherently dependent, introducing additional complexity to the analysis. This dependence presents two key challenges. First, under PS scheduling, later arrivals joining the queue of the tagged task influence its response time, which due to the interdependence between the two queues, requires tracking the queue length at both queues throughout the process. Second, the absence of a product-form joint queue-length distribution complicates the deconditioning step, which, unlike in previous cases where it could be obtained by simply evaluating a generating function at ρ_1, ρ_2 , is no longer straightforward.

We reuse the results of [Har24], where the generating function of the queue-length distribution was obtained. We then derive two functional equations: one for the generating function of the LST of the conditional response time of a task joining queue 1 and another for a task joining queue 2. These equations are significantly more complicated than those in the PS-PS case, as they contain partial generating functions as well as their derivatives. We use the term *partial generating function* because they are power series in two variables that are not summed over the full first quadrant $\{(i, j) \mid 0 \leq i < \infty, 0 \leq j < \infty\}$. Despite this added complexity, we successfully obtain solutions using the approximation method. Additionally, we derive a recursive method for calculating the response time moments for a task joining either queue 1 or queue 2. To address the deconditioning step, we evaluate a certain complex integral in two dimensions to obtain the LST of unconditional response time density and its moments. Finally, the LST is inverted using the Fixed Talbot method as before, and the resulting density function and first four moments are compared against regenerative simulation, showing high accuracy.

The rest of the chapter is organized as follows. The functional equations for the generating functions of conditional LSTs are obtained in Section 4.2. The functional equations are transformed into a problem in linear algebra and subsequently solved in Section 4.3. The method is evaluated in Section 4.4. Two optimisation methods are proposed and evaluated in Section 4.5. A summary of the main contributions of the chapter is presented in Section 4.6.

4.2 Obtaining the functional equations

As outlined in Section 2.4.2, the queueing model under consideration consists of two parallel queues each with exponential service times and service rates of μ_1 and μ_2 , respectively. Both queues use PS discipline locally. Arrivals are Poisson with rate λ , assuming $\lambda < \mu_1 + \mu_2$ for stability, and join the shortest queue. If at an arrival instant, the queues are of equal length, the arriving job is routed to queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. Figure 4.1 displays the above network at an arrival instant when queue 1 has length $i = 6$ and queue 2 has length $j = 4$. The arriving task is therefore joining queue 2.

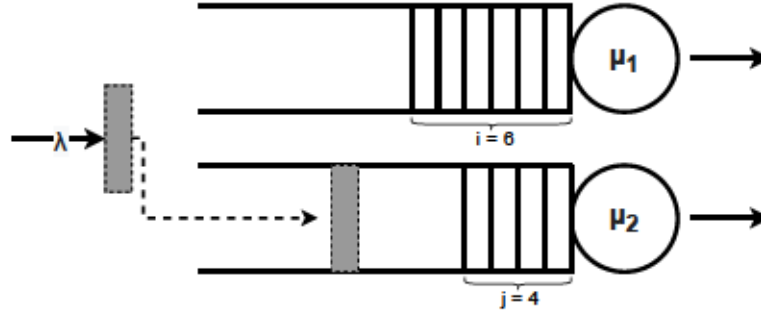


Figure 4.1: Two JSQ-PS queues in parallel where the arriving task joins the shorter queue. The arrival rate is λ , the service rates are μ_1 and μ_2 ,

The goal is to derive the LST of the response time probability distribution, which is then inverted numerically to obtain the response time density. In order to do that, we first derive two functional equations. These functional equations are then solved in Section 4.3.

To deal with response times at equilibrium, one first needs to obtain the joint steady-state queue-length distribution. This is derived for JSQ servers with FCFS queueing discipline in [Har24], but PS has exactly the same balance equations for the queue length probabilities in a Markovian model, and hence the same solution. Let $G(x, y)$ denote the generating function of the steady state queue-length probabilities, that is $G(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \pi_{i,j} x^i y^j$ where $\pi_{i,j}$ is the probability of i and j customers in the first and second queue, respectively, at equilibrium. From this point onward, we assume $G(x, y)$ is known.

4.2.1 Generating functions of conditional response time distributions

Our first step is to obtain functional equations for the generating functions of the LST of response time distributions, conditioned on the number of tasks in the first and second queue at the time of the arrival of the tagged task.

Let the random variables $U_{i,j}$, for $i, j \geq 0$, denote the response time of a particular customer joining queue 1 when there are i and j other customers present in queues 1 and 2, respectively. Let $U_{i,j}$ have probability distribution function $U_{i,j}(t)$, with LST $U_{i,j}^*(s)$. We define $V_{i,j}, V_{i,j}(t)$ and $V_{i,j}^*(s)$ analogously for a tagged customer joining queue 2. We denote the generating

function of $U_{i,j}^*(s)$ by $E(x, y, s)$ and the generating function of $V_{i,j}^*(s)$ by $F(x, y, s)$.

In order to derive functional equations for $E(x, y, s)$ and $F(x, y, s)$, in Definition 4.1, we define certain related Taylor series, each holomorphic in unit disks centred at the origin. Some of them are termed “partial generating functions” because they are power series in two variables that are not summed over the full first quadrant $\{(i, j) \mid 0 \leq i < \infty, 0 \leq j < \infty\}$. We use the following notation for functions $f(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{i,j} x^i y^j$ and for binary operator $\sim$:

$$f^{\sim} = \sum_{i \sim j} a_{i,j} x^i y^j$$

Note that $f^{\sim}$, where $\sim$ stands for one of $<, >, =$, can be expressed using $f^{\leq}(x, y), f^{\geq}(x, y)$ and $f(x, y)$. For example, $f^{=}(x, y) = f(x, y)^{\leq} + f(x, y)^{\geq} - f(x, y)$.

Definition 4.1. *Generating functions corresponding to $U_{i,j}^*(s)$ and $V_{i,j}^*(s)$ and related analytic functions:*

1. *Generating functions*

$$E(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^*(s) x^i y^j \quad F(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} V_{i,j}^*(s) x^i y^j$$

2. *Partial generating functions*

$$E^{\leq}(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=i}^{\infty} U_{i,j}^*(s) x^i y^j \quad E^{\geq}(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^i U_{i,j}^*(s) x^i y^j$$

Analogous expressions are defined for $F^{\leq}, F^{\geq}$, replacing U by V on the right-hand side.

3. *Related analytic functions*

$$\begin{aligned} \alpha_E(x, y, s) &:= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i+1,j}^*(s) x^i y^j & \gamma_E(x, y, s) &:= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i+2,j}^*(s) x^i y^j \\ \beta_E(x, y, s) &:= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j+1}^*(s) x^i y^j & \delta_E(x, y, s) &:= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j+2}^*(s) x^i y^j \end{aligned}$$

and $\alpha_F(x, y, s), \beta_F(x, y, s), \gamma_F(x, y, s), \delta_F(x, y, s)$ are defined by replacing U by V on the right hand sides.

4. Partial versions of related analytic functions

$$\alpha_E^{\leq}(x, y, s) = \sum_{i=0}^{\infty} \sum_{j=i}^{\infty} U_{i+1,j}^*(s) x^i y^j \quad \alpha_E^{\geq}(x, y, s) = \sum_{i=0}^{\infty} \sum_{j=0}^i U_{i+1,j}^*(s) x^i y^j$$

Similarly for $\beta_E^{\leq}, \beta_E^{\geq}, \gamma_E^{\leq}, \gamma_E^{\geq}, \delta_E^{\leq}, \delta_E^{\geq}$ and for $\alpha_F, \beta_F, \gamma_F, \delta_F$ by replacing U by V .

The following lemma shows how the related analytic functions can be expressed in terms of the original generating functions E and F . It is easy to derive by appropriate change of variable over the summation domains.

Lemma 4.2.1.

$$\begin{aligned} \alpha_E(x, y, s) &= \frac{1}{x} (E(x, y, s) - E(0, y, s)) \\ \beta_E(x, y, s) &= \frac{1}{y} (E(x, y, s) - E(x, 0, s)) \\ \gamma_E(x, y, s) &= \frac{1}{x^2} \left(E(x, y, s) - E(0, y, s) - x \frac{\partial E}{\partial x}(0, y, s) \right) \\ \delta_E(x, y, s) &= \frac{1}{y^2} \left(E(x, y, s) - E(x, 0, s) - y \frac{\partial E}{\partial y}(x, 0, s) \right) \end{aligned}$$

Analogous expressions can be obtained for $\alpha_F, \beta_F, \gamma_F, \delta_F$, replacing E by F on the right-hand sides.

Partial generating functions can be computed as follows [Har24],

$$f^{\leq}(x_0, y_0) = \frac{1}{2\pi i} \oint_{C_y} \frac{f(x_0 y_0 / y, y)}{y - y_0} dy$$

where $f(x, y)$ is a two-parameter function that is analytic for all x, y with $|x| \leq r_1, |y| \leq r_2$. Furthermore, $|x_0| < r_1, |y_0| < r_2$ and the contour C_y is the circle with radius r_2 . Again, a similar expression can be derived for $f^{\geq}$.

We now have the following result for the generating functions E and F corresponding to the LST of conditional response time distributions for a task joining queue 1 and queue 2, respectively.

Proposition 4.2.1. *The generating functions $E(x, y, s)$ and $F(x, y, s)$ are given by the following equations*

$$\begin{aligned} & (s + \lambda + (1 - x)\mu_1 + (1 - y)\mu_2) \left(E(x, y, s) + x \frac{\partial E}{\partial x}(x, y, s) \right) - \mu_2 \left(E(x, 0, s) + x \frac{\partial E}{\partial x}(x, 0, s) \right) \\ & - \lambda \frac{\partial E^<}{\partial x}(x, y, s) - \lambda \left(\beta_E^>(x, y, s) + x \frac{\partial \beta_E^>}{\partial x}(x, y, s) \right) - \lambda a_1 \frac{\partial E^=}{\partial x}(x, y, s) \\ & - \lambda a_2 y \left(\delta_E^=(x, y, s) + x \frac{\partial \delta_E^=}{\partial x}(x, y, s) \right) = \frac{\mu_1}{(1 - x)(1 - y)} \end{aligned} \quad (4.1)$$

$$\begin{aligned} & (s + \lambda + (1 - x)\mu_1 + (1 - y)\mu_2) \left(F(x, y, s) + y \frac{\partial F}{\partial y}(x, y, s) \right) - \mu_1 \left(F(0, y, s) + y \frac{\partial F}{\partial y}(0, y, s) \right) \\ & - \lambda \frac{\partial F^>}{\partial y}(x, y, s) - \lambda \left(\alpha_F^<(x, y, s) + y \frac{\partial \alpha_F^<}{\partial y}(x, y, s) \right) - \lambda a_2 \frac{\partial F^=}{\partial y}(x, y, s) \\ & - \lambda a_1 x \left(\gamma_F^=(x, y, s) + y \frac{\partial \gamma_F^=}{\partial y}(x, y, s) \right) = \frac{\mu_2}{(1 - x)(1 - y)} \end{aligned} \quad (4.2)$$

Proof. Suppose that at time t , the tagged customer is in queue 1 with i other customers and there are j customers at queue 2. Then, in the infinitesimal interval $(t, t + h]$ there are a number of possible events, one of which occurs with non-zero probability to first order in h :

1. An arrival to queue 1, if $i + 1 < j$, with probability λh ;
2. An arrival to queue 2, if $i + 1 > j$, with probability λh ;
3. An arrival to queue 1, if $i + 1 = j$, with probability $a_1 \lambda h$;
4. An arrival to queue 2, if $i + 1 = j$, with probability $a_2 \lambda h$;
5. A departure from queue 2, if $j > 0$, with probability $\mu_2 h$;
6. A departure from queue 1 of a customer other than the tagged customer, if $i > 0$, with probability $\frac{i}{i+1} \mu_1 h$;
7. Completion of the sojourn of the tagged customer with probability $\frac{1}{i+1} \mu_1 h$;
8. No event occurs with probability $1 - (\lambda + \mu_1 + \mu_2 I_{j>0})h$

We therefore have the following equation to first order in h for $i, j \geq 0$, with terms ordered according to the above list of events.

$$\begin{aligned} U_{i,j}(t + h) = & h \lambda I_{i+1 < j} U_{i+1,j}(t) + h \lambda I_{i+1 > j} U_{i,j+1}(t) + h \lambda a_1 I_{i+1 = j} U_{i+1,j}(t) + h \lambda a_2 I_{i+1 = j} U_{i,j+1}(t) \\ & + h \mu_2 I_{j > 0} U_{i,j-1}(t) + h \mu_1 I_{i > 0} \frac{i}{i+1} U_{i-1,j}(t) + \frac{h \mu_1}{i+1} U_{i,j}(t) + (1 - h(\lambda + \mu_1 + \mu_2 I_{j > 0})) U_{i,j}(t) \end{aligned}$$

where I is the indicator function, i.e. I_B is 1 if B is true and 0 otherwise. Rearranging, dividing by h and taking the limit $h \rightarrow 0$ then yields

$$\begin{aligned} \frac{dU_{i,j}}{dt} = & \lambda I_{i+1 < j} U_{i+1,j}(t) + \lambda I_{i+1 > j} U_{i,j+1}(t) + \lambda a_1 I_{i+1=j} U_{i+1,j}(t) + \lambda a_2 I_{i+1=j} U_{i,j+1}(t) \\ & + \mu_2 I_{j>0} U_{i,j-1}(t) + \mu_1 I_{i>0} \frac{i}{i+1} U_{i-1,j}(t) + \frac{\mu_1}{i+1} - (\lambda + \mu_1 + \mu_2 I_{j>0}) U_{i,j}(t) \end{aligned}$$

Taking the LST and multiplying throughout by $(i+1)$ we get

$$\begin{aligned} (s + \lambda + \mu_1 + \mu_2)(i+1)U_{i,j}^*(s) - \mu_2 I_{j=0}(i+1)U_{i,0}^*(s) = & \lambda I_{i+1 < j}(i+1)U_{i+1,j}^*(s) \\ & + \lambda I_{i+1 > j}(i+1)U_{i,j+1}^*(s) + \lambda a_1 I_{i+1=j}(i+1)U_{i+1,j}^*(s) + \lambda a_2 I_{i+1=j}(i+1)U_{i,j+1}^*(s) \\ & + \mu_2 I_{j>0}(i+1)U_{i,j-1}^*(s) + \mu_1 I_{i>0} i U_{i-1,j}^*(s) + \mu_1 \end{aligned}$$

where $U_{i,j}^*(s)$ is the LST of the distribution function $U_{i,j}(t)$. Multiplying by $x^i y^j$ and summing over $0 \leq i, j < \infty$, with appropriate changes to these summation variables, yields (dropping the argument s for brevity)

$$\begin{aligned} (s + \lambda + \mu_1 + \mu_2) & \left(x \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^* i x^{i-1} y^j + \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^* x^i y^j \right) - \mu_2 \left(x \sum_{i=0}^{\infty} U_{i,0}^* i x^{i-1} + \sum_{i=0}^{\infty} U_{i,0}^* x^i \right) \\ = & \lambda \sum_{i=1}^{\infty} \sum_{j=i+1}^{\infty} U_{i,j}^* i x^{i-1} y^j + \lambda \left(x \sum_{i=0}^{\infty} \sum_{j=0}^i U_{i,j+1}^* i x^{i-1} y^j + \sum_{i=0}^{\infty} \sum_{j=0}^i U_{i,j+1}^* x^i y^j \right) + \lambda a_1 \sum_{i=1}^{\infty} U_{i,i}^* i x^{i-1} y^i \\ & + \lambda a_2 y \left(x \sum_{i=0}^{\infty} U_{i,i+2}^* i x^{i-1} y^i + \sum_{i=0}^{\infty} U_{i,i+2}^* x^i y^i \right) + \mu_2 y \left(x \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^* i x^{i-1} y^j + \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^* x^i y^j \right) \\ & + \mu_1 x \left(x \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^* i x^{i-1} y^j + \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U_{i,j}^* x^i y^j \right) + \mu_1 \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} x^i y^j \end{aligned}$$

Plugging in the definitions for $E(x, y)$, $E^\sim(x, y)$, $\beta_E^\sim(x, y)$ and $\delta_E^\sim(x, y)$ and moving everything but the last term to the left-hand side, we get (4.1) as required.

The proof of (4.2) follows the exact same steps; it is omitted for brevity. $\square$

Note that the above equations have at most one solution due to the uniqueness of the steady-state distribution in the underlying Markov process. So far, we have derived functional equa-

tions for E in (4.1) and F in (4.2). Our next task is to combine these generating functions in order to express the LST of unconditional response time distribution.

4.2.2 LST of the unconditional response time distribution

The functional equations (4.1) and (4.2) are solved for generating functions E and F in Section 4.3. However, it is still not straightforward to express the LST of unconditional response time distributions in terms of the above generating functions.

We already noted that the queue-length distribution does not have a product form, however, in the case of FCFS scheduling at the individual queues, the LST of response time distribution is readily expressed from just the generating function of queue-length probabilities $G(x, y)$ as follows [Har24]

$$\begin{aligned} W^*(s) = & a_2 \left(\frac{\mu_1}{s + \mu_1} \right) [G(\mu_1/(s + \mu_1), 1) - G^{\geq}(\mu_1/(s + \mu_1), 1)] + \\ & a_1 \left(\frac{\mu_2}{s + \mu_2} \right) [G(1, \mu_2/(s + \mu_2)) - G^{\leq}(1, \mu_2/(s + \mu_2))] + \\ & a_1 \left(\frac{\mu_1}{s + \mu_1} \right) G^{\leq}(\mu_1/(s + \mu_1), 1) + a_2 \left(\frac{\mu_2}{s + \mu_2} \right) G^{\geq}(1, \mu_2/(s + \mu_2)) \end{aligned}$$

This is because, under the FCFS discipline, later arrivals do not impact the response time of the tagged task. However, as discussed in Section 2.4, this is not the case for the PS discipline, where overtaking can occur. Additionally, due to the interdependence between the queue lengths at the two queues, both queues must be tracked throughout the calculation of the response time.

Thus, calculating the response time distribution under PS is notably more complicated than under FCFS. The conditional LSTs are not geometric expressions, meaning additional work is required for deconditioning. The necessary Laplace-Stieltjes transform is presented in the following proposition, where a carefully selected complex integral ensures that the correct indices remain from the quadrupole sum obtained by multiplying the generating function of the conditional LSTs with the generating function of the queue-length probabilities.

Proposition 4.2.2. *The Laplace transform of response time density in a pair of $M/M/1$ queues with JSQ-PS scheduling, omitting s from the integrands for brevity, is*

$$\begin{aligned}
W^*(s) = \frac{1}{(2\pi)^2} & \left(\int_0^{2\pi} \int_0^{2\pi} E^<(r_1 e^{it_1}, r_1 e^{it_2}) G(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 + \right. \\
& \int_0^{2\pi} \int_0^{2\pi} F^>(r_1 e^{it_1}, r_1 e^{it_2}) G(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 + \\
& \int_0^{2\pi} \int_0^{2\pi} a_1 E^=(r_1 e^{it_1}, r_1 e^{it_2}) G(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 + \\
& \left. \int_0^{2\pi} \int_0^{2\pi} a_2 F^=(r_1 e^{it_1}, r_1 e^{it_2}) G(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 \right) \quad (4.3)
\end{aligned}$$

where $r_1 \leq 1, r_2 = \frac{1}{r_1} \leq \frac{\mu_1 + \mu_2}{\lambda}$ and i is the imaginary unit.

Proof.

$$\begin{aligned}
W^*(s) &= \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} (I_{i<j} U_{i,j}^*(s) + I_{i>j} V_{i,j}^*(s) + I_{i=j} a_1 U_{i,j}^*(s) + I_{i=j} a_2 V_{i,j}^*(s)) \pi_{i,j} \\
&= \sum_{i<j} U_{i,j}^*(s) \pi_{i,j} + \sum_{i>j} V_{i,j}^*(s) \pi_{i,j} + a_1 \sum_{i=j} U_{i,j}^*(s) \pi_{i,j} + a_2 \sum_{i=j} V_{i,j}^*(s) \pi_{i,j}
\end{aligned}$$

Omitting s for brevity and using the notation " $\sim$ " as before for a binary operator, we focus on the $U_{i,j}^*$ terms first. In order to be able to express $\sum_{i \sim j} U_{i,j}^* \pi_{i,j}$, we need to extract coefficients that satisfy $i = i'$ and $j = j'$ from the sum

$$E^{\sim}(x, y) G(z, q) = \sum_{i \sim j} \sum_{i'=0}^{\infty} \sum_{j'=0}^{\infty} U_{i,j}^* \pi_{i',j'} x^{i'} y^{j'} z^{i'} q^{j'}$$

The trick is a carefully chosen complex integral that filters out unwanted terms from the product of the generating functions. This can be achieved by applying Lemma 1 in [HB21a], which is stated in a more generic setting. Here we provide an alternative, shorter argument for our specific case. Given $r_1 \leq 1, r_2 = 1/r_1 \leq \frac{\mu_1 + \mu_2}{\lambda}$, we have

$$\begin{aligned}
& \frac{1}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} E^{\sim}(r_1 e^{it_1}, r_1 e^{it_2}) G(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 \\
&= \frac{1}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} \sum_{i \sim j} \sum_{i'=0}^{\infty} \sum_{j'=0}^{\infty} U_{i,j}^* \pi_{i',j'} r_1^{i'} r_2^{j'} (e^{it_1})^{i-i'} r_1^j r_2^{j'} (e^{it_2})^{j-j'} dt_1 dt_2 \\
&= \sum_{i \sim j} \sum_{i'=0}^{\infty} \sum_{j'=0}^{\infty} U_{i,j}^* \pi_{i',j'} r_1^{i+j} r_2^{i'+j'} \left(\frac{1}{2\pi} \int_0^{2\pi} (e^{it_1})^{i-i'} dt_1 \right) \left(\frac{1}{2\pi} \int_0^{2\pi} (e^{it_2})^{j-j'} dt_2 \right) \\
&= \sum_{i \sim j} U_{i,j}^* \pi_{i,j} (r_1 r_2)^{i+j} = \sum_{i \sim j} U_{i,j}^* \pi_{i,j}
\end{aligned}$$

where for the penultimate equality, we used the fact that

$$\int_0^{2\pi} e^{itk} dt = \begin{cases} 0 & \text{if } k \neq 0 \\ 2\pi & \text{if } k = 0 \end{cases}$$

The sums containing the $V_{i,j}^*$ terms are obtained similarly. □

In this procedure, the choices of r_1 and r_2 are important. The generating functions of the conditional Laplace transforms ($E(x, y, s)$ and $F(x, y, s)$) diverge at $r_1 = 1$ when $s = 0$, each then being equal to $1/((1-x)(1-y))$. Similarly, $G(x, y)$ diverges at its radius of convergence, which we took to be $(\mu_1 + \mu_2)/\lambda$. We chose a value for r_2 halfway between 1 and the radius of convergence of G , i.e. $r_2 = \frac{\lambda + \mu_1 + \mu_2}{2\lambda}$. However, note that for heavily loaded systems, $\lambda \simeq \mu_1 + \mu_2$ so that r_1 ends up near 1 and r_2 near the radius of convergence of G , which can lead to numerical errors at very high loads. Nonetheless, the approximation remains quite accurate even at $\approx 83\%$ utilisation, as shown in Section 4.4.

4.3 Solving the functional equations

In Section 4.2, we stated three functional equations for the generating functions of conditional and unconditional response time distribution LSTs, these are (4.1), (4.2), and (4.3). To solve these functional equations numerically, we transform them into a problem in linear algebra by following the steps detailed for one dimension in Section 3.3.2 and for two dimensions in Section 3.4. Beyond the matrix operations listed in Table 3.14, the functional equations introduce additional operations, one of which is partialization. The matrices representing partialization were obtained in [Har24] by considering each point $x_1, \dots, x_n, y_1, \dots, y_n$ in turn, distorting the contour to go around it, and applying Cauchy's Integral Theorem. For convenience, we provide an updated table of operations, including the matrices for partialization, shown in Table 4.1. Notably, the matrices for differentiation and partialization are constant, applying to all functional equations with n discretization points in both complex planes.

Table 4.1: Matrices representing operations in two dimensions.

Operation	Matrix
Identity operator giving f	$I^{nm \times nm}$
Multiplication by known function $A(x, y)$	$\Lambda(\vec{a})$
Differentiation w.r.t. x	$M_{1,0}$
Differentiation w.r.t. y	$M_{0,1}$
Evaluation at an internal point $u_0 \in D_x$	U_{u_0}
Evaluation at an internal point $v_0 \in D_y$	V_{v_0}
Partialization giving $f^{\leq}$	$P^{\leq}$
Partialization giving $f^{\geq}$	$P^{\geq}$

Table 4.2: Matrices representing the operations corresponding to the related analytic functions of Definition 4.1.

Operation	Matrix
$\alpha_f(x, y) = \frac{1}{x} [f(x, y) - f(0, y)]$	$A^{nm \times nm}$
$\beta_f(x, y) = \frac{1}{y} [f(x, y) - f(x, 0)]$	$B^{nm \times nm}$
$\gamma_f(x, y) = \frac{1}{x^2} \left[f(x, y) - f(0, y) - x \frac{\partial}{\partial x} f(0, y) \right]$	$C^{nm \times nm}$
$\delta_f(x, y) = \frac{1}{y^2} \left[f(x, y) - f(x, 0) - y \frac{\partial}{\partial y} f(x, 0) \right]$	$D^{nm \times nm}$

4.3.1 Conditional response times

In order to transform the equations for functions $E(x, y, s)$ and $F(x, y, s)$, we need to define further matrices representing the related analytic functions given in Definition 4.1. The following proposition derives these matrices. The operations together with their corresponding matrix representations are shown in Table 4.2. Again, assuming the discretisation points are fixed, the corresponding matrices in Table 4.2 are all constant, applying to any functional equation solved for the same set of discretisation points.

Proposition 4.3.1. *Define the following operators associated with function $f(x, y)$:*

$$\begin{aligned}
 \alpha_f(x, y) &= \frac{1}{x} [f(x, y) - f(0, y)] & \gamma_f(x, y) &= \frac{1}{x^2} \left[f(x, y) - f(0, y) - x \frac{\partial}{\partial x} f(0, y) \right] \\
 \beta_f(x, y) &= \frac{1}{y} [f(x, y) - f(x, 0)] & \delta_f(x, y) &= \frac{1}{y^2} \left[f(x, y) - f(x, 0) - y \frac{\partial}{\partial y} f(x, 0) \right]
 \end{aligned}$$

and let A, B, C and D be the corresponding matrix representations, respectively, given by:

$$\begin{aligned} A &= \Lambda(1/\vec{x})(I^{n^2 \times n^2} - U_0) \\ B &= \Lambda(1/\vec{y})(I^{n^2 \times n^2} - V_0) \\ C &= \Lambda(1/\vec{x}^2)(I^{n^2 \times n^2} - U_0 - \Lambda(\vec{x})U_0M_{1,0}) \\ D &= \Lambda(1/\vec{y}^2)(I^{n^2 \times n^2} - V_0 - \Lambda(\vec{y})V_0M_{0,1}) \end{aligned}$$

Equipped with these new matrices, the transformation of the equations for $E(x, y, s)$ and $F(x, y, s)$, albeit more complicated, follows the same procedure introduced in Section 3.4. For the vector representation $\vec{e}$ of $E(x, y, s)$, by replacing operators with their corresponding matrices in Table 4.1 and Table 4.2, we obtain from (4.1)

$$\begin{aligned} & \left(\Lambda(s + \lambda + (1 - \vec{x})\mu_1 + (1 - \vec{y})\mu_2)(I^{n^2 \times n^2} + \Lambda(\vec{x})M_{1,0}) - \mu_2(V_0 + \Lambda(\vec{x})V_0M_{1,0}) \right. \\ & \quad - \lambda M_{1,0}(I^{n^2 \times n^2} - P^{\geq}) - \lambda(I^{n^2 \times n^2} + \Lambda(\vec{x})M_{1,0})P^{\geq}B \\ & \quad \left. - \lambda a_1 M_{1,0}P^= - \lambda a_2 \Lambda(\vec{y})(I^{n^2 \times n^2} + \Lambda(\vec{x})M_{1,0})P^=D \right) \vec{e} \\ & = \left(\frac{\mu_1}{(1 - x_i)(1 - y_j)} \mid 1 \leq i \leq n, 1 \leq j \leq n \right) \end{aligned} \quad (4.4)$$

where the vector on the right-hand side is ordered lexicographically as usual. Similarly, for the vector representation $\vec{f}$ of $F(x, y, s)$, we obtain

$$\begin{aligned} & \left(\Lambda(s + \lambda + (1 - \vec{x})\mu_1 + (1 - \vec{y})\mu_2)(I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1}) - \mu_1(U_0 + \Lambda(\vec{y})U_0M_{0,1}) \right. \\ & \quad - \lambda M_{0,1}(I^{n^2 \times n^2} - P^{\leq}) - \lambda(I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1})P^{\leq}A \\ & \quad \left. - \lambda a_2 M_{0,1}P^= - \lambda a_1 \Lambda(\vec{x})(I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1})P^=C \right) \vec{f} \\ & = \left(\frac{\mu_2}{(1 - x_i)(1 - y_j)} \mid 1 \leq i \leq n, 1 \leq j \leq n \right) \end{aligned} \quad (4.5)$$

Introducing the notation

$$\vec{c} = \left(\frac{1}{(1 - x_i)(1 - y_j)} \mid 1 \leq i \leq n, 1 \leq j \leq n \right) \quad (4.6)$$

where the column vector on the right-hand side is ordered lexicographically as usual, we can now approximate the function $E(x, y, s)$ by an interpolation of the vector representation

$$\vec{e} = \Gamma_E(s) \mu_1 \vec{c}$$

of $E(x, y, s)$ over the discretization set pairs $\{(x_i, y_j) \mid 1 \leq i, j \leq n\}$, where

$$\begin{aligned} \Gamma_E(s) = & \left(\Lambda(s + \lambda + (1 - \vec{x})\mu_1 + (1 - \vec{y})\mu_2) (I^{n^2 \times n^2} + \Lambda(\vec{x})M_{1,0}) \right. \\ & - \mu_2 (V_0 + \Lambda(\vec{x})V_0M_{1,0}) \\ & - \lambda M_{1,0} (I^{n^2 \times n^2} - P^{\geq}) - \lambda (I^{n^2 \times n^2} + \Lambda(\vec{x})M_{1,0}) P^{\geq} B - \lambda a_1 M_{1,0} P^= \\ & \left. - \lambda a_2 \Lambda(\vec{y}) (I^{n^2 \times n^2} + \Lambda(\vec{x})M_{1,0}) P^= D \right)^{-1} \end{aligned} \quad (4.7)$$

Similarly, $F(x, y, s)$ is approximated by an interpolation of the vector representation

$$\vec{f} = \Gamma_F(s) \mu_2 \vec{c}$$

of $F(x, y, s)$ over the discretization set pairs $\{(x_i, y_j) \mid 1 \leq i, j \leq n\}$, where

$$\begin{aligned} \Gamma_F(s) = & \left(\Lambda(s + \lambda + (1 - \vec{x})\mu_1 + (1 - \vec{y})\mu_2) (I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1}) \right. \\ & - \mu_1 (U_0 + \Lambda(\vec{y})U_0M_{0,1}) \\ & - \lambda M_{0,1} (I^{n^2 \times n^2} - P^{\leq}) - \lambda (I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1}) P^{\leq} A - \lambda a_2 M_{0,1} P^= \\ & \left. - \lambda a_1 \Lambda(\vec{x}) (I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1}) P^= C \right)^{-1} \end{aligned} \quad (4.8)$$

4.3.2 Unconditional response time

Define the vector representation of the partial versions of the generating function E and F as

$$\begin{aligned} \vec{e}^< &= (I - P^{\geq}) \vec{e} & \vec{e}^= &= P^= \vec{e} \\ \vec{f}^> &= (I - P^{\leq}) \vec{f} & \vec{f}^= &= P^= \vec{f} \end{aligned}$$

Then (4.3) for $W^*(s)$ in Proposition 4.2.2 can be transformed as follows, omitting s from the integrands for brevity,

$$W^*(s) \approx \frac{1}{(2\pi)^2} \left(\int_0^{2\pi} \int_0^{2\pi} \mathfrak{J}(\vec{e}^<)(r_1 e^{it_1}, r_1 e^{it_2}) \mathfrak{J}(\vec{g})(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 + \right. \\ \int_0^{2\pi} \int_0^{2\pi} \mathfrak{J}(\vec{f}^>)(r_1 e^{it_1}, r_1 e^{it_2}) \mathfrak{J}(\vec{g})(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 + \\ \int_0^{2\pi} \int_0^{2\pi} a_1 \mathfrak{J}(\vec{e}^=)(r_1 e^{it_1}, r_1 e^{it_2}) \mathfrak{J}(\vec{g})(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 + \\ \left. \int_0^{2\pi} \int_0^{2\pi} a_2 \mathfrak{J}(\vec{f}^=)(r_1 e^{it_1}, r_1 e^{it_2}) \mathfrak{J}(\vec{g})(r_2 e^{-it_1}, r_2 e^{-it_2}) dt_1 dt_2 \right)$$

where $\mathfrak{J}(\vec{e}^\sim)$, $\mathfrak{J}(\vec{f}^\sim)$ and $\mathfrak{J}(\vec{g})$ are two-variable interpolations of $\vec{e}^\sim$, $\vec{f}^\sim$ and $\vec{g}$, respectively, defined above. We use linear interpolation.

One can choose the contours on which the original discretisation points lie to match the circles of this double integral that calculates $W^*(s)$ according to (4.3). That is, the circles used in the estimation of E and F have radius r_1 , and the circle used to estimate the probability generating function G has radius $r_2 = 1/r_1$. This leads to a straightforward evaluation of the integral by interpolating the discretised values around the circles.

4.3.3 Moments

As we have seen before, the k th moment of response time is $(-1)^k$ multiplied by the k th derivative of $W^*(s)$ evaluated at $s = 0$. This requires the partial derivatives of $E(x, y, s)$ and $F(x, y, s)$ with respect to s at $s = 0$, which are determined below in Lemma 4.3.1. First, recall that the solution for the vector representation of $E(x, y, s)$ in (4.4) is $\vec{e} = \Gamma_E(s) \mu_1 \vec{c}$. The results for $F(x, y, s)$ are similar.

Lemma 4.3.1. *The k th partial derivative of $E(x, y, s)$ with respect to s at $s = 0$ has vector representation $\vec{e}_k$ given by the recurrence*

$$\vec{e}_0 = \Gamma_E(0) \mu_1 \vec{c} \\ \vec{e}_k = -k \Gamma_E(0) \left(\mathbf{I}^{n^2 \times n^2} + \Lambda(\vec{x}) \mathbf{M}_{1,0} \right) \vec{e}_{k-1}$$

where $\vec{c}$ is given by (4.6) and $\Gamma_E(0)$ is defined by (4.7) above with the substitution $s = 0$.

Similarly, the k th partial derivative of $F(x, y, s)$ with respect to s at $s = 0$ has vector representation $\vec{f}_k$ given by the recurrence

$$\begin{aligned}\vec{f}_0 &= \Gamma_F(0)\mu_2\vec{c} \\ \vec{f}_k &= -k\Gamma_F(0) \left(I^{n^2 \times n^2} + \Lambda(\vec{y})M_{0,1} \right) \vec{f}_{k-1}\end{aligned}$$

where $\vec{c}$ is the same as before and $\Gamma_F(0)$ is defined by (4.8) above with the substitution $s = 0$.

Proof. First, as already noted, $\vec{e}_0 = \Gamma_E(0)\mu_1\vec{c}$. Differentiating (4.1) k times with respect to s (denoted by the superscript (k)), noting that the variable s appears in only one coefficient and that the right-hand side is constant, yields, for $k \geq 1$,

$$\begin{aligned}& (s + \lambda + (1-x)\mu_1 + (1-y)\mu_2) \left(E^{(k)}(x, y, s) + x \frac{\partial E^{(k)}}{\partial x}(x, y, s) \right) \\ & - \mu_2 \left(E^{(k)}(x, 0, s) + x \frac{\partial E^{(k)}}{\partial x}(x, 0, s) \right) - \lambda \frac{\partial E^{(k)<}}{\partial x}(x, y, s) \lambda (\beta_E^{(k)\geq}(x, y, s) + x \frac{\partial \beta_E^{(k)\geq}}{\partial x}(x, y, s)) \\ & - \lambda a_1 \frac{\partial E^{(k)=}}{\partial x}(x, y, s) - \lambda a_2 y (\delta_E^{(k)=}(x, y, s) + x \frac{\partial \delta_E^{(k)=}}{\partial x}(x, y, s)) \\ & = -k \left(E^{(k-1)}(x, y, s) + x \frac{\partial E^{(k-1)}}{\partial x}(x, y, s) \right)\end{aligned}$$

The left-hand side is the left-hand side of (4.1) with E replaced by $E^{(k)}$. Hence, if we replace $\vec{c}$ with

$$\vec{c}_k = \left(-k \left(E^{(k-1)}(x_i, y_j, 0) + x_i \frac{\partial E^{(k-1)}}{\partial x}(x_i, y_j, 0) \right) \mid 1 \leq i \leq n, 1 \leq j \leq n \right)$$

we find that $\vec{e}_k = \Gamma_E(0)\vec{c}_k$. The result then follows for $\vec{e}_k$ using the operators in Table 4.1. The proof for $\vec{f}_k$ is similar. $\square$

We note that there is no need to recalculate the inverse of a matrix for each moment; the same inverse matrix $\Gamma_E(0)$ or $\Gamma_F(0)$ is all that is needed. As a result, once these inverses have been computed, any number of moments can be calculated quickly. The unconditional response time moments are obtained by deconditioning according to Proposition 4.2.2, replacing $W^*(0)$ by

Table 4.3: Response time moments for two JSQ-PS queues with parameters $\lambda = 1$, $\mu_1 = 1$, $\mu_2 = 2$, $a_1 = 1$, $a_2 = 0$.

Moment	JSQ/PS model	Simulation	95% CB
1st	0.959	0.959	± 0.003
2nd	2.263	2.270	± 0.023
3rd	9.253	9.304	± 0.207
4th	54.984	55.494	± 2.402

the k th moment M_k , E by $E^{(k)}$ and F by $F^{(k)}$; actually, for $k = 0$, $W^*(0) = M_0 = 1$, providing an accuracy check.

4.4 Evaluation

In order to evaluate the methodology presented above for estimating the response time distribution of two JSQ-PS queues, we perform several experiments. First, the numerical response time moments and density are obtained for three networks, corresponding to a light, a medium and a heavy load scenario. The results are compared against regenerative simulation, following the methods set out in Section 3.4.3. Next, JSQ scheduling is compared against static scheduling methods. This is followed by a section devoted to finding the optimal value of a_1 , the probability of a customer joining the first queue given that the queues are of equal length. We then assess the accuracy and computational cost of the algorithm. Finally, two optimisations are considered and evaluated to further enhance the accuracy of the algorithm.

4.4.1 Comparison with simulation

We first evaluate our model by calculating response time densities for three sets of parameters. These parameters are chosen to test the model in light, medium, and heavy load scenarios. The expectation is to have higher accuracy in the case of light and medium loads. We invert the Laplace transforms numerically using the Fixed Talbot inversion method as before.

For the light load scenario, the parameters used are $\lambda = 1$, $\mu_1 = 1$, $\mu_2 = 2$, $a_1 = 1$, $a_2 = 0$, giving $\approx 33\%$ utilisation; when the queue lengths are equal, arrivals are sent to the slower queue.

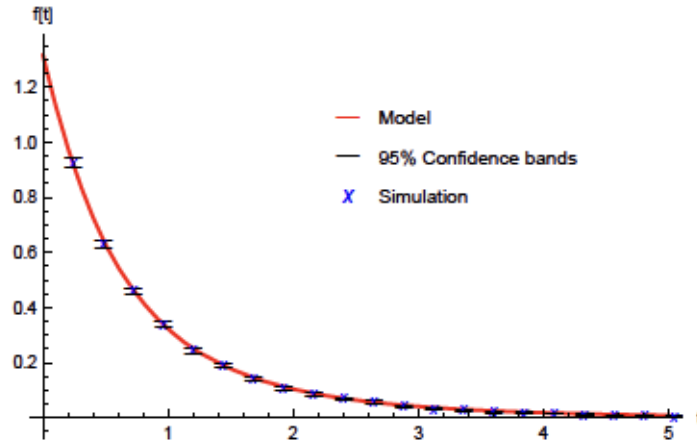


Figure 4.2: Response time density for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 1, \mu_2 = 2, a_1 = 1, a_2 = 0$; comparison with simulation.

We use 32 points around the circle for both coordinates when solving for E and F , as under light load the generating functions are smoother, and therefore fewer points are required to achieve highly accurate results. The density obtained is then plotted against the corresponding simulation results. For the latter, we use regenerative simulation with 250,000 regeneration cycles. Figure 4.2 shows that the response time density calculated by the model is extremely accurate in this case. Next, we use Lemma 4.3.1 to compute response time moments up to the fourth by differentiation of $W^*(s)$ at $s = 0$ using $n = 64$ discretisation points. This is much more efficient than calculating the response time densities as it only requires the model to run for a single s -value, namely $s = 0$. This is in contrast to estimating the density, where, at each time-point t , we need to calculate $W^*(s)$ for at least 16 s -values in order to be able to invert the LST with reasonable accuracy. Table 4.3 compares the first four moments of response time to those obtained from the corresponding regenerative simulation in the case of light traffic. The moments calculated by the model are very accurate, falling well within the 95% confidence bands.

For the medium load scenario, the parameters used are $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$, giving 50% utilisation. Notice that in this case all arrivals are directed to the faster server when the queue lengths are equal. We revert to using 64 points around the circle for both coordinates to solve for E and F when calculating the density and the first four moments to achieve good accuracy. Note that, compared with the $M/M/1$ -PS and PS-PS models from the

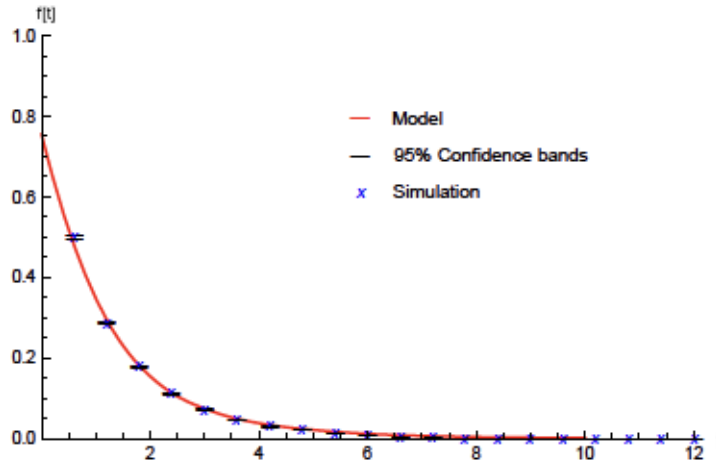


Figure 4.3: Response time density for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$; comparison with simulation.

Table 4.4: Response time moments for two JSQ-PS queues with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$.

Moment	JSQ/PS model	Simulation	95% CB
1st	1.396	1.390	± 0.007
2nd	4.574	4.528	± 0.068
3rd	25.719	25.355	± 0.887
4th	215.229	213.849	± 16.047

previous chapters, JSQ-PS is much more complex. Although the asymptotic accuracy of the approximation method is the same across all three problems, in practice, there is a significant increase in running time and memory requirements compared with the simpler models. Figure 4.3 shows that the accuracy of the density calculated by the model is still remarkable. The moments corresponding to this scenario are shown in Table 4.4, and they match the simulated values very closely, falling within the 95% confidence bands.

For the heavy traffic scenario, the parameters used are $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$ — giving $\approx 83\%$ utilisation — so either queue would be saturated on its own since $\lambda > \mu_2 > \mu_1$; the scheduling strategy is therefore more critical. As before, we compare the results with the corresponding simulation. The densities are shown in Figure 4.4 and still show good agreement. Note, that the numbers of points used to calculate E and F were increased from 64 to 100 to achieve sufficient accuracy. Finally, we once again use Lemma 4.3.1 to compute response time

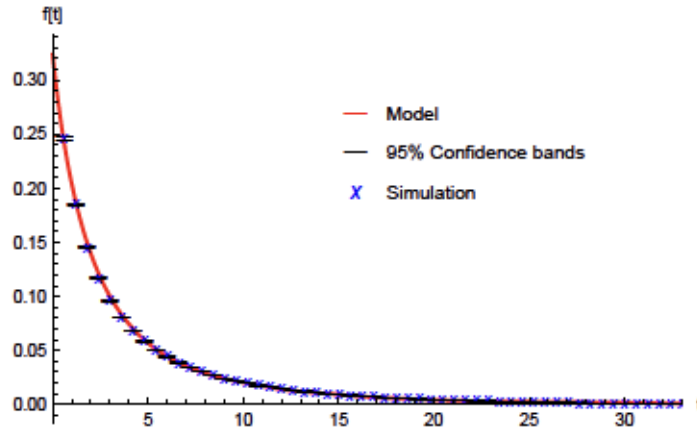


Figure 4.4: Response time density for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$; comparison with simulation.

Table 4.5: Response time moments for two JSQ-PS queues with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$

Moment	JSQ/PS model	Simulation	95% CB
1st	6.155	6.091	± 0.023
2nd	125.622	124.991	± 1.310
3rd	5472.74	5547.67	± 128.6
4th	403965	422517	± 19383

moments. Table 4.5 compares the first four moments of response time to those obtained from the corresponding regenerative simulation in the case of heavy traffic. The model results show very good accuracy for higher moments, well within the 95% confidence intervals, and the first moment just outside.

4.4.2 Static scheduling models

It is interesting to compare this JSQ model with other, static scheduling policies for the same two servers with total arrival rate λ . The obvious policy would be to split the arrival stream so that each queue gets arrivals at a rate proportional to its service rate; i.e. arrival rates $p_1\lambda$ and $p_2\lambda$ to queue 1 and queue 2, respectively, where $p_1 = \mu_1/(\mu_1 + \mu_2)$ and $p_2 = \mu_2/(\mu_1 + \mu_2)$. The two queues then have known response time densities [CJMT70]. Notice that not any choice of probabilities p_1, p_2 give equilibrium, which requires $p_1\lambda < \mu_1$ and $p_2\lambda < \mu_2$; i.e. $1 - \mu_2/\lambda < p_1 < \mu_1/\lambda$. Hence in our higher load case, we require $0.2 < p_1 < 0.4$. The load-balanced choice

Table 4.6: Comparison of response time moments for JSQ/PS, JSQ/FCFS, static load balanced with PS and static load balanced with FCFS. The parameters are $\lambda = 1$, $\mu_1 = 0.4$, $\mu_2 = 0.8$, $a_1 = a_2 = 0.5$.

Moment	JSQ/PS	JSQ/FCFS	Static/PS	Static/FCFS
1st	6.15533	6.07415	10.0000	10.0000
2nd	125.622	79.398	385.714	225.000
3rd	5472.74	1678.18	35127.60	8437.50
4th	403965	49942	5862020	455625

above has $p_1 = \mu_1/(\mu_1 + \mu_2) = 1/3$, $p_2 = 2/3$ and yields equal utilisation 0.8333 at the two queues. The response time in the static scheduling model has first four moments (obtained by differentiating the known Laplace transform of the density function) 10,385.714, 35,127.6, and 5,862,020, very much higher than those achieved by JSQ scheduling. This is expected, as JSQ scheduling drives the system toward equilibrium at each arrival instant, thereby limiting growth in either queue and consequently reducing the higher moments compared with the case where the queues are considered in isolation.

4.4.3 Finding the optimal routing probabilities

In this section, our goal is to find the value of a_1 —the probability of an arriving task joining queue 1 given that the queues have equal length—that produces the best performance, that is, one that corresponds to the lowest moments. In the case of two identical queues ($\mu_1 = \mu_2$), it is straightforward to see that the optimal routing is $a_1 = a_2 = 0.5$, otherwise, one of the servers becomes slightly overloaded. However, when the service rates are different, $a_1 = a_2 = 0.5$ is not expected to be optimal. One obvious candidate to try is splitting arrivals in proportion to the service rate of each queue, similar to the static scheduling model above, making $a_1 = \frac{\mu_1}{\mu_1 + \mu_2}$.

The set of parameters used for this experiment is $\lambda = 2$, $\mu_1 = 1$, $\mu_2 = 3$, and we take several values of a_1 between 0 and 1. The results are displayed in Table 4.7. The columns of Table 4.7 correspond to different values of a_1 and we calculate the first four moments for each a_1 value. The a_1 value corresponding to the lowest moments is $a_1 = 0$. This suggests, perhaps not surprisingly, that when the service rates differ widely—in the current example queue 2 serves

Table 4.7: Moments as the probability, a_1 , of joining queue 1 at equal queue lengths varies; The parameters are $\lambda = 2, \mu_1 = 1, \mu_2 = 3$.

Moments	$a_1 = 0.0$	$a_1 = 0.2$	$a_1 = 0.4$	$a_1 = 0.6$	$a_1 = 0.8$	$a_1 = 1.0$
1st	0.731	0.787	0.839	0.885	0.927	0.969
2nd	1.638	1.929	2.214	2.488	2.749	3.009
3rd	7.455	9.458	11.524	13.607	15.681	17.794
4th	53.204	71.357	90.948	111.481	132.634	154.705

Table 4.8: Moments as the probability, a_1 , of joining queue 1 at equal queue lengths varies. The parameters are $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$.

Moments	$a_1 = 0.0$	$a_1 = 0.2$	$a_1 = 0.4$	$a_1 = 0.6$	$a_1 = 0.8$	$a_1 = 1.0$
1st	1.396	1.406	1.434	1.452	1.469	1.486
2nd	4.574	4.640	4.818	4.951	5.094	5.247
3rd	25.719	26.332	27.765	29.019	30.43	32.021
4th	215.229	222.872	238.088	253.074	270.534	290.419

customers three times as fast as queue 1—the optimal strategy is to send everything to the faster server when there is a choice.

We rerun the experiments with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$ to see if this is still the case when the service rates are much closer together; now queue 2 is only slightly more efficient than queue 1. Table 4.8 shows the results. The first row corresponding to $a_1 = 0$ remains the one with the lowest moments, suggesting that sending everything to the faster server is still the optimal strategy, even when service rates differ ever so slightly. As shown in [GBSW07], OPT-0 — a scheduling policy that chooses the queue minimising the mean response time of all jobs currently in the system, assuming no future arrivals — is a more optimal strategy for parallel PS queues than JSQ. While it is generally true that when the queues are of equal length the faster server would be chosen by OPT-0, suggesting the faster server as an obvious choice for tie breaking in JSQ, there remain situations in which OPT-0 may assign the job to the slower server.

These findings prompt us to think about variations of JSQ scheduling itself. One can imagine a scheduling similar to JSQ where, given $\mu_1 < \mu_2$, arriving customers go to queue 1 if $i < j - 1$

and to queue 2 if $i > j - 1$, and routing probabilities are used to break ties when $i = j - 1$. More generally, we can shift the difference in the number of jobs by k , and route customers to the slower server when its queue is k customers shorter than the queue of the faster server. A natural question to ask is what is the optimal value of k for a given pair of service rates.

4.4.4 Accuracy and cost

As per [Har24], we know the worst-case errors for each of the matrices given in Table 4.1. The worst-case errors of Table 4.2 are covered by the same propositions, since using Proposition 4.3.1, the matrices can be expressed as linear combinations of matrices in Table 4.1. The operation with the largest asymptotic error corresponds to calculating derivatives with a worst-case error of $\Theta(n^{-1})$ for each element of the matrix. Once the functional equations are transformed into a system of linear equations, the matrix on the left-hand side is inverted. In our numerical calculations, the condition number of this matrix is low so that the error of the inverted matrix remains of the same order. Finally, the inverted matrix is multiplied with a vector that is known exactly. Therefore, the error of the sum is of order $\sqrt{n}\Theta(n^{-1}) = \Theta(n^{-1/2})$ where the multiple of $\sqrt{n}$ comes from the Central Limit Theorem and $\Theta(n^{-1})$ is the individual error of each term. However, we found the error to be around 10-100 times smaller in practice; this is unsurprising because the above error estimates correspond to the worst-case scenario, not the average.

Regarding the cost of the algorithm, each of the matrices in tables 4.1 and 4.2, corresponding to the operators in (4.4) and (4.5), has size $n^2 \times n^2$ and is calculated element-wise. Therefore, each requires storage and has computation time of order $\mathcal{O}(n^4)$. We used approximately 100 points in each contour, so this is in the region of 10^8 . However, these constant matrices need only be calculated once for any given n and then can be stored and reused. To get the final equations of form $\mathbf{A}(s)\vec{x} = \vec{b}$, a linear combination of the individual matrices in Table 4.1 needs to be calculated, which depends on s , but these calculations are much faster than the calculations of the matrices in Table 4.1. However, for every time-point t in the density function, we need approximately 16 – 32 values of s , and corresponding matrices $\mathbf{A}(s)$, to be able to invert the Laplace transform numerically. The product of 32 and the number of time points required

to approximate the density function well is significantly larger than n in our calculations, so the overall cost exceeds $\Theta(n^5)$. Next, the linear equations need to be solved for each s -point, which requires the inversion of the matrices $A(s)$ – a cubic operation in the dimension of the input matrix, in our case giving $\mathcal{O}(n^6)$ cost. This is the dominating factor, but in practice, we found the time needed to be of the same order as calculating the $A(s)$ matrices. This may be partly because $A(s)$ requires the calculation of several matrices that each take $\Theta(n^5)$ time. The remaining operations, e.g., inverting the Laplace transforms to obtain the densities and the calculation of moments, are negligible in comparison. Optimisation of these calculations is possible and would likely decrease the computational cost several-fold. Finally, note that the corresponding simulation results have run for several days.

4.5 Optimisations

In this section, two possible optimisation methods are considered to increase the accuracy of our algorithm.

4.5.1 Optimisation I: inversion of operator order

The first optimisation inverts the order of some of the operators in equations (4.1) and (4.2). These equations have several terms involving a derivative of a partial generating function. We evaluate the efficacy of performing the differentiation first and then transforming the resulting generating functions into partial versions. This could improve accuracy as the generating functions are Laplace transforms – smooth functions whose derivatives, when approximated, are likely to retain high precision. In fact when $s = 0$, as in the calculation of moments, the Laplace transforms are the constant function 1, with easily approximated derivative 0 everywhere. To facilitate this inversion, we need the following auxiliary lemma.

Lemma 4.5.1. *If $H(x, y)$ is a two-parameter function that is analytic for all x, y with $|x| \leq r_1, |y| \leq r_2$, then for any x_0, y_0 with $|x_0| = r_1$ and $|y_0| = r_2$*

$$H^>(x_0, y_0) = -\frac{1}{2\pi i} \oint_{C_y} \frac{H(x_0 y_0 / y, y)}{y - y_0} dy \quad (4.9)$$

$$H^<(x_0, y_0) = -\frac{1}{2\pi i} \oint_{C_x} \frac{H(x, x_0 y_0/x)}{x - x_0} dx \quad (4.10)$$

where the contours C_x, C_y are, respectively, circles centred on the origin and lying in the annuli of analyticity of $H(x, x_0 y_0/x), H(x_0 y_0/y, y)$, with radii less than r_1, r_2 .

Proof. Let $H(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij} x^i y^j$ with radii of convergence R_x, R_y , so that $r_1 < R_x$ and $r_2 < R_y$. Then the annulus of analyticity of $H(x_0 y_0/y, y)$ is the region between the circles $D(0, r_1 r_2/R_x)$ and $D(0, r_2)$. Noting that $|y| < |y_0|$ for $y \in C_y$, we have

$$\begin{aligned} \oint_{C_y} \frac{H(x_0 y_0/y, y)}{y - y_0} dy &= - \oint_{C_y} \frac{H(x_0 y_0/y, y)}{y_0} \frac{1}{1 - y/y_0} dy \\ &= - \oint_{C_y} \left[\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij} y_0^{-1} (x_0 y_0)^i y^{j-i} \right] \left[\sum_{n=0}^{\infty} (y/y_0)^n \right] dy \\ &= - \oint_{C_y} \left[\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \sum_{n=0}^{\infty} a_{ij} y_0^{-1} (x_0 y_0)^i y^{j-i+n} y_0^{-n} \right] dy \\ &= - \oint_{C_y} \left[\sum_{i=0}^{\infty} \sum_{j=0}^{i-1} a_{ij} y_0^{-1} (x_0 y_0)^i y^{-1} y_0^{j+1-i} \right] dy = -2\pi i H^>(x_0, y_0) \end{aligned}$$

Equation(4.10) follows by symmetry. □

Remark. For (4.9) (respectively (4.10)), the integration contour lies in the annulus of convergence of $H(x_0 y_0/y, y)$ (respectively $H(x, x_0 y_0/x)$) and encircles the origin, passing inside the point y_0 . Since $H(x_0 y_0/y, y)$ and $H(x, x_0 y_0/x)$ are analytic in the annuli of convergence, the Cauchy-integral around a small closed contour γ , say, oriented counter-clockwise, that encloses the points y_0, x_0 in the respective cases, each evaluates to $2\pi i H(x_0, y_0)$. Consequently, since $H^<(x_0, y_0) + H^>(x_0, y_0) = H(x_0, y_0)$, application of Cauchy's integral theorem to the counter-clockwise oriented "inner" contour C_y joined onto γ , which converts C_y into an "outer" contour, yields the result by application of Lemma 2 of [Har24].

Corollary 4.5.1. *Lemma 4.5.1 extends to the case where the contours C_x, C_y are, respectively, the circles centred on the origin with radii r_1, r_2 , distorted to go around the points x_0, y_0 on the inner side¹.*

¹i.e. the distortions (typically semi-circles with vanishing radius) traverse the point on the side towards the origin.

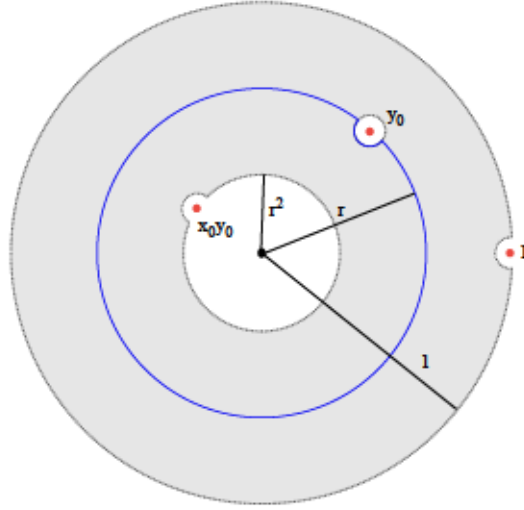


Figure 4.5: The contour of integration (in blue) is the distorted circle going around y_0 on the inner side, and the area of convergence (in grey) is the punctured annulus with inner radius r^2 and outer radius 1.

The proof follows by continuity since in the region $D(0, r_1) \times D(0, r_2)$ and on its boundaries ($x = r_1$ or $y = r_2$), $H(x, y)$ is analytic.

The following result now provides the necessary formulas for inverting the order of differentiation and partialization.

Proposition 4.5.1. *If $H(x, y)$ is a two-parameter function that is analytic for all x, y with $|x| \leq r, |y| \leq r$, then for any x_0, y_0 with $|x_0| = |y_0| = r$*

$$\begin{aligned} \frac{\partial}{\partial x} H^>(x_0, y_0) &= \left(\frac{\partial}{\partial x} H(x_0, y_0) \right)^{\geq} \\ \frac{\partial}{\partial x} H^<(x_0, y_0) &= \left(\frac{\partial}{\partial x} H(x_0, y_0) \right)^{<} \\ \frac{\partial}{\partial x} H^=(x_0, y_0) &= \frac{1}{x_0} \left(x_0 \frac{\partial}{\partial x} H(x_0, y_0) \right)^= \end{aligned}$$

$\frac{\partial}{\partial x} H^{\geq}(x_0, y_0)$ and $\frac{\partial}{\partial x} H^{<}(x_0, y_0)$ can be derived from the above, using the linearity of differentiation, e.g., $\frac{\partial}{\partial x} H^{\geq}(x_0, y_0) = \frac{\partial}{\partial x} H^>(x_0, y_0) + \frac{\partial}{\partial x} H^=(x_0, y_0)$.

Proof. Using Lemma 4.5.1 and its corollary, we have

$$\begin{aligned}\frac{\partial}{\partial x} H^>(x_0, y_0) &= \frac{\partial}{\partial x} \left(-\frac{1}{2\pi i} \oint_{C_y} \frac{H(x_0 y_0 / y, y)}{y - y_0} dy \right) \\ &= -\frac{1}{2\pi i} \oint_{C_y} \frac{\frac{\partial}{\partial x} H(x_0 y_0 / y, y)}{y - y_0} \frac{y_0}{y} dy\end{aligned}$$

where the contour of integral C_y shown in Figure 4.5 goes around y_0 on the inner side.

Now, let $x = x_0 y_0 / y$ and note that this change of variable transforms C_y into a clockwise contour C_x that goes around x_0 on the outside. Therefore, we get

$$\frac{1}{2\pi i} \oint_{C_x} \frac{\frac{\partial}{\partial x} H(x, x_0 y_0 / x)}{x_0 y_0 / x - y_0} \frac{y_0 x}{x_0 y_0} \left(-\frac{x_0 y_0}{x^2} \right) dx = \frac{1}{2\pi i} \oint_{C_x} \frac{\frac{\partial}{\partial x} H(x, x_0 y_0 / x)}{x - x_0} dx$$

This is exactly the integral given by [Har24, Lemma 2] for $(\frac{\partial}{\partial x} H(x_0, y_0))^{\geq}$. The other two equations can be proved following similar steps. $\square$

Similar equations can be derived for partial derivatives with respect to y_0 exploiting symmetry. In matrix form, Proposition 4.5.1 is applied by, for example, substituting $M_{1,0} P^=$ with $\Lambda(1/\vec{x}) P^= \Lambda(\vec{x}) M_{1,0}$ in the definition of $\Gamma_E(s)$ in (4.7).

4.5.2 Optimisation II: simplification of the partialization operator

Similar to the PS-PS tandem queue in Section 3.3, when evaluated at $s = 0$, the generating functions simplify to $E(x, y, 0) = F(x, y, 0) = \frac{1}{(1-x)(1-y)}$. We proceed analogously by defining

$$\tilde{E}(x, y, s) = (1-x)(1-y)E(x, y, s)$$

and

$$\tilde{F}(x, y, s) = (1-x)(1-y)F(x, y, s)$$

Consequently, we have $\tilde{E}(x, y, 0) = \tilde{F}(x, y, 0) = 1$ smoothing out the sharp peaks of E and F near $x = y = 1$. We now derive the matrix-vector equations for $\tilde{E}(x, y, s)$ and $\tilde{F}(x, y, s)$ by adapting Proposition 4.2.1; however, the transformation of the equations is not as straightforward as in the PS-PS case due to their complexity. Therefore, instead of rewriting the equations

directly, we define linear mappings between the two vector spaces as follows.

Note that $\tilde{E}(x, y, s) = (1-x)(1-y)E(x, y, s)$ is a linear transformation of $E(x, y, s)$. In vector-matrix notation, $\tilde{\vec{e}} = \Lambda((1-\bar{x})(1-\bar{y}))\vec{e}$. This defines a linear map $\mathcal{V} \rightarrow \tilde{\mathcal{V}}$ on the vector space $\mathcal{V}$ of complex vectors of length n^2 . Then, each operator matrix M of (4.4) and (4.5) in $\mathcal{V}$ is mapped to the matrix $\tilde{M} = \Lambda((1-\bar{x})(1-\bar{y}))M\Lambda(\frac{1}{(1-\bar{x})(1-\bar{y})})$ in $\tilde{\mathcal{V}}$. The equation we solve in $\tilde{\mathcal{V}}$, with mapped vectors and matrices, remains the same. Our aim now is to calculate and simplify the matrices for some of the operations in the new vector space directly. For differentiation and evaluation at $x = 0$ or $y = 0$, this is straightforward. For example,

$$\tilde{M}_{1,0} = \Lambda((1-\bar{x})(1-\bar{y}))M_{1,0}\Lambda\left(\frac{1}{(1-\bar{x})(1-\bar{y})}\right) = \Lambda\left(\frac{1}{1-\bar{x}}\right) + M_{1,0}$$

and

$$\tilde{U}_0 = \Lambda((1-\bar{x})(1-\bar{y}))U_0\Lambda\left(\frac{1}{(1-\bar{x})(1-\bar{y})}\right) = \Lambda(1-\bar{y})U_0$$

To calculate the matrices for partialization in the new vector space, we derive equations in the following lemma that relate the partialized versions of generating function $\tilde{E}$ to those of E .

Lemma 4.5.2. *Let $r = r_1 = r_2 < 1$ and $|x_0| = |y_0| = r$. Then*

$$E^>(x_0, y_0) = \frac{x_0 \tilde{E}^<(1, x_0 y_0)}{(x_0 - 1)(x_0 y_0 - 1)} + \frac{\tilde{E}^>(x_0, y_0)}{(x_0 - 1)(y_0 - 1)} - \frac{\tilde{E}^>(x_0 y_0, 1)}{(y_0 - 1)(x_0 y_0 - 1)} \quad (4.11)$$

$$E^<(x_0, y_0) = \frac{y_0 \tilde{E}^>(x_0 y_0, 1)}{(y_0 - 1)(x_0 y_0 - 1)} + \frac{\tilde{E}^<(x_0, y_0)}{(x_0 - 1)(y_0 - 1)} - \frac{\tilde{E}^<(1, x_0 y_0)}{(x_0 - 1)(x_0 y_0 - 1)} \quad (4.12)$$

$$E^=(x_0, y_0) = \frac{\tilde{E}^=(x_0, y_0) + \tilde{E}^>(x_0 y_0, 1) + \tilde{E}^<(1, x_0 y_0)}{1 - x_0 y_0} \quad (4.13)$$

Proof. Let C_y be the circle centred on the origin with radius r_2 , distorted to go round y_0 on the inner side as shown in Figure 4.5. Using Lemma 4.5.1 and its corollary, we have

$$E^>(x_0, y_0) = \frac{1}{2\pi i} \oint_{C_y} \frac{\tilde{E}(x_0 y_0 / y, y)}{(1 - x_0 y_0 / y)(y - 1)(y - y_0)} dy = \frac{1}{2\pi i} \oint_{C_y} \frac{y \tilde{E}(x_0 y_0 / y, y)}{(y - x_0 y_0)(y - 1)(y - y_0)} dy$$

Noting that y_0, x_0y_0 and 1 are three distinct values we can decompose the integrand into partial fractions. Omitting the repeating $\tilde{E}(x_0y_0/y, y)$ factors for brevity, we have

$$\begin{aligned} \frac{y}{(y - x_0y_0)(y - 1)(y - y_0)} &= \frac{x_0}{(x_0 - 1)(x_0y_0 - 1)} \frac{1}{y - x_0y_0} \\ &\quad - \frac{1}{(x_0 - 1)(y_0 - 1)} \frac{1}{y - y_0} + \frac{1}{(y_0 - 1)(x_0y_0 - 1)} \frac{1}{y - 1} \end{aligned}$$

The first of the ensuing three integrals is given by Lemma 2 of [Har24] as x_0y_0 lies inside the contour C_y , whereas the other two integrals are calculated using Lemma 4.5.1 as y_0 and 1 both lie outside. Hence

$$\begin{aligned} \frac{1}{2\pi i} \oint_{C_y} \frac{y\tilde{E}(x_0y_0/y, y)}{(y - x_0y_0)(y - 1)(y - y_0)} dy &= \frac{x_0}{(x_0 - 1)(x_0y_0 - 1)} \tilde{E}^{\leq}(1, x_0y_0) \\ &\quad + \frac{1}{(x_0 - 1)(y_0 - 1)} \tilde{E}^>(x_0, y_0) - \frac{1}{(y_0 - 1)(x_0y_0 - 1)} \tilde{E}^>(x_0y_0, 1) \end{aligned}$$

Equation (4.12) follows by symmetry and (4.13) by using $E^= = E - E^> - E^<$, noting that $E^=(x_0, y_0) = E^=(1, x_0y_0) = E^=(x_0y_0, 1)$. $\square$

Using Lemma 4.5.2, matrix $\mathbf{P}^=$ in the new vector space, for example, becomes

$$\begin{aligned} \tilde{\mathbf{P}}^= &= \Lambda((1 - \vec{x})(1 - \vec{y})) \mathbf{P}^= \Lambda \left(\frac{1}{(1 - \vec{x})(1 - \vec{y})} \right) \\ &= \Lambda \left(\frac{(1 - \vec{x})(\vec{1} - \vec{y})}{1 - \vec{x}\vec{y}} \right) (\mathbf{P}^= + \mathbf{P}_{xy,1}^> + \mathbf{P}_{1,xy}^<) \end{aligned}$$

where $\mathbf{P}_{xy,1}^>$ is the matrix corresponding to the linear approximation of the operation

$$H^>(x_0y_0, 1) = -\frac{1}{2\pi i} \int_{C_y} \frac{H(x_0y_0/y, y)}{y - 1} dy$$

4.5.3 Numerical experiments

We evaluate the optimisations presented above on three queueing models: two operating under medium load (50% utilisation), as shown in tables 4.9 and 4.10, and one with heavy load ($\sim 83\%$ utilisation), displayed in Table 4.11. At medium load, we find 64 discretisation points

Table 4.9: Comparison of the optimised methods with the original algorithm and simulation for calculating response time moments (model parameters: $\lambda = 2, \mu_1 = 1, \mu_2 = 3$ and $a_1 = 0.5$.)

Moment	JSQ/PS	Opt. I	Opt. I + II	Simulation	95% CB
1st	0.863	0.861	0.860	0.857	± 0.004
2nd	2.353	2.337	2.336	2.316	± 0.028
3rd	12.565	12.418	12.413	12.237	± 0.314
4th	101.119	99.374	99.449	97.211	± 4.629

to be sufficient on both the x - and y -contours, but for the high load parameterization 64 points give poor accuracy, and we use 120. For all three models, two rounds of testing are carried out. In the first round, *Optimisation I* is run by itself. The results are shown in the third column of tables 4.9, 4.10, and 4.11. There are improvements in accuracy across all moments compared to the original algorithm, achieved without any increase in the running time. It is important to note, however, that in the case of heavy load, the simulation results — especially the higher moments — are prone to error, and we believe that the moments calculated by our model provide more reliable estimates.

Motivated by these findings, we ran *Optimisation I* and *Optimisation II* together in the second round of testing. The results are displayed in the fourth column of the tables. While combining the two optimisation methods further enhances the accuracy of the results, this improvement comes at the cost of a significant increase in running time. In the case of medium load, the gains compared to using *Optimisation I* alone are relatively minor. In the case of high load, the first moment now falls within the 95% confidence band, making running the combined optimisation more justified. Therefore, we conclude that *Optimisation I* is the most effective approach for most networks as it increases the accuracy without adding to the running time, and the error in the first moment in the case of heavy traffic when compared to the combined optimisation is only 0.59%.

Table 4.10: Comparison of the optimised methods with the original algorithm and simulation for calculating response time moments (model parameters: $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$ and $a_1 = 0.5$.)

Moment	JSQ/PS	Opt. I	Opt. I + II	Simulation	95% CB
1st	1.443	1.441	1.440	1.438	± 0.007
2nd	4.883	4.871	4.866	4.863	± 0.067
3rd	28.371	28.266	28.228	28.278	± 0.853
4th	245.269	244.102	243.676	243.700	± 14.042

Table 4.11: Comparison of the optimised methods with the original algorithm and simulation for calculating response time moments (model parameters: $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8$ and $a_1 = 0.5$.)

Moment	JSQ/PS	Opt. I	Opt. I + II	Simulation	95% CB
1st	6.155	6.127	6.108	6.091	± 0.023
2nd	125.622	126.327	125.966	124.991	± 1.310
3rd	5472.74	5597.28	5629.87	5547.67	± 128.6
4th	403965	423419	434455	422517	± 19383

4.6 Summary

In this chapter, we have derived the response time density in a pair of PS queues with JSQ scheduling, for the first time. Our method accommodates heterogeneous service rates and arbitrary routing probabilities, representing the most general form of JSQ-PS scheduling with two parallel queues. The main contributions of the chapter are as follows:

- The functional equations for the generating functions E and F of conditional response times are derived, where the conditioning is on whether the arriving tagged task joins the first or the second queue. These functional equations are then solved numerically by transforming them into a problem in linear algebra.
- Once the generating functions G, E, F are approximated, where G is the generating function of steady-state queue-length probabilities obtained in [Har24], together with their modifications (e.g. $E^\sim$), we evaluate a complex integral to express the LST of the uncon-

ditional response time distribution, from which the density itself is obtained by numerical inversion. Moments are obtained very efficiently in a straightforward manner by recursively calculating derivatives of $W^*(s)$ at $s = 0$

- We demonstrate that our numerical method accurately computes the response time density and corresponding moments under both heavy and medium loads. Furthermore, we find the optimal routing probabilities and compare JSQ-PS scheduling to static scheduling methods.
- Finally, we propose two optimisation methods to increase the accuracy of the method, which are evaluated for several sets of parameters.

Chapter 5

A Compositional PH fitting algorithm

5.1 Introduction

In this chapter, we first propose a novel fitting technique to estimate the service PH distribution in a PH/PH/1 queue. Surveys such as [ANT21] indicate a gap in the literature for estimating arrival and service distributions tailored to QBD processes. Unlike the case of directly fitting a PH distribution to a trace of inter-arrival times or service times, inverse problems involving QBD processes fitted from queueing data have rarely been considered [KML05]. The present work seeks to address this gap by focusing on fitting the QBD service process, leveraging knowledge of both the arrival process and the marginal distribution of queue-length probabilities, either measured or simulated. That is, we seek to fit PH distributions through the inverse problem that matches QBD steady-state formulas to empirical values of marginal queue-length probabilities. This ensures that once fitted, the PH distributions account for the observed system behaviour, conducing to parameterizations that better match empirical observations also when changing the characteristics of the underlying queueing system.

To make our technique effective, we focus on queues with compositional, KPC-PH service processes. We prove the surprising result that in such a queue, the solution of the QBD process can be *decomposed* into the solution of independent, but much smaller, QBD processes. The approach is either exact or approximate depending on the characteristics of the arrival process.

While divide-and-conquer approaches are already available to search a space of PH distributions for fitting purposes (e.g., [CZS10a]), to the best of our knowledge, this is the first work that develops a PH fitting method of this kind efficiently searching over a space of QBD processes. In terms of computational gains, our KPC-QBD method reduces the time complexity of calculating marginal queue-length probabilities in a QBD from cubic to quadratic. In practice, execution times that take up to a few minutes using the original QBD are now reduced to seconds with the decomposed QBD. Moreover, PH distributions of very large size, over a thousand states, can be effectively handled as inputs to QBD processes with our method, whereas they are normally impractical to use in QBD analysis. Combining the fast calculation of queue-length probabilities with numerical search methods, we develop a fitting algorithm that can efficiently explore service distributions that result in QBD processes that accurately fit the empirical queueing data.

Validation experiments on real traces reveal that our method can learn service distributions from queueing data more accurately than existing methods that fit the service distributions directly. The increased accuracy also holds when calculating response time distributions for the same queueing model. Furthermore, the fitted PH models display low prediction errors when used to parameterise queueing models, which are different from those they are originally fitted on, showing low generalisation errors.

Motivated by the accuracy of the fitted distributions produced by the above method, next, we briefly explore fitting ME distributions to traces directly. We generalise the construction of KPC-PH distributions — used as the service process during the QBD fitting — to produce a special family of ME distributions that are valid by construction, addressing a key challenge in their use. This extension preserves the compositional aspects of KPC-PH distributions, which are particularly beneficial for fitting. Specifically, the elements of this new family of distributions allow for efficient calculation of the CDF and Mellin transforms. Numerical results show that a fitting algorithm driven by these measures utilising the constructed family of ME distributions performs extremely well when applied to fitting real-life traces directly.

The remainder of the chapter is organized as follows. Section 5.2 presents a motivating example. Section 5.3 develops the decomposition result for solving QBD processes. Section 5.4 applies

Table 5.1: Summary of the main notation

<i>Symbol</i>	<i>Description</i>
N, N, n	N is a PH KPC component with state space N , cardinality n
J	number of hyperexponential components in the KPC process
$H_k, H_k, h_k, k = 1, \dots, J$	H_k is a hyperexponential KPC component with state space H_k and cardinality h_k
S, S, s	$S = N \otimes H_1 \otimes \dots \otimes H_J$ is a KPC distribution with state space S and cardinality s
A, A, a	A is a PH arrival process with state space A and cardinality a
$A/S/1$	a KPC-queue with arrival process A and KPC service process
λ_A^*	mean arrival rate of process A
(A, N, H, L)	state space of a KPC-queue with $J = 1$, L denotes the level of the queue
m	number phases in the QBD on each level
$\mu_N^*, \mu_{H_k}^*$	mean service rate of process N and H_k , respectively
N^*, H_k^*	scaled version of N and H_k , respectively
CQ_N	component queue $A/N^*/1$
CQ_{H_k}	component queue $M/H_k^*/1$, M is Poisson and has arrival rate 1
$\mathbf{1}_n$	column vector of ones with length n
I_n	identity matrix of size $n \times n$

the results of the previous section to QBD process fitting. In Section 5.5, we report and discuss our numerical results. Section 5.6 introduces a novel ME fitting algorithm. Finally, a summary of the achievements of the chapter is given in Section 5.7.

5.2 Motivating example

This example illustrates the decomposition properties that arise naturally when using QBD processes with KPC-PH service distribution. We consider a simple $M/PH/1$ queue where the service time distribution is a KPC-PH distribution composed of a two-phase Erlang distribution and a two-phase hyperexponential distribution. Using matrix notation, the exponential arrival process $A = (A_0, A_1)$ is simply

$$A_0 = -\lambda \quad A_1 = \lambda$$

with $\lambda \geq 0$. For the service process, we use the Kronecker product composition of a two-phase Erlang distribution $\mathbf{N} = (\mathbf{N}_0, \mathbf{N}_1)$,

$$\mathbf{N}_0 = \begin{bmatrix} -\mu & \mu \\ 0 & -\mu \end{bmatrix} \quad \mathbf{N}_1 = \begin{bmatrix} 0 & 0 \\ \mu & 0 \end{bmatrix}$$

with $\mu > 0$, and a two-phase hyperexponential distribution $\mathbf{H} = (\mathbf{H}_0, \mathbf{H}_1)$,

$$\mathbf{H}_0 = \begin{bmatrix} -\gamma_1 & 0 \\ 0 & -\gamma_2 \end{bmatrix} \quad \mathbf{H}_1 = \begin{bmatrix} p\gamma_1 & (1-p)\gamma_1 \\ p\gamma_2 & (1-p)\gamma_2 \end{bmatrix}$$

with $\gamma_1 \geq \gamma_2 > 0$ and $0 < p < 1$. The KPC-PH service process is therefore represented as $\mathbf{S} = (\mathbf{S}_0, \mathbf{S}_1)$ with $\mathbf{S}_0 = -\mathbf{N}_0 \otimes \mathbf{H}_0$ and $\mathbf{S}_1 = \mathbf{N}_1 \otimes \mathbf{H}_1$. In this queue, the matrices defining the QBD process are readily computed as

$$\mathbf{B} = \mathbf{S}_1, \quad \mathbf{L} = -\lambda \mathbf{I}_2 + \mathbf{S}_0, \quad \mathbf{F} = \lambda \mathbf{I}_2.$$

Let $\vec{\alpha}$ denote the equilibrium probability vector for the service process embedded at departure instants; that is, $\vec{\alpha} = \mathbf{1}^\top (-\mathbf{S}_0)^{-1} \mathbf{S}_1$. In the case of an M/PH/1 queue, matrix $\mathbf{R}$ in (2.5) is known explicitly [Neu82] and can be calculated as

$$\mathbf{R} = -\lambda(-\lambda \mathbf{I}_2 + \mathbf{S}_0 + \lambda \mathbf{1} \vec{\alpha})^{-1}$$

We also know that, in this case, due to the Poisson arrival process, $\mathbf{G} = \mathbf{1} \vec{\alpha}$ [LR99]. This can be easily verified using the probabilistic interpretation of matrix $\mathbf{G}$ given in (2.7).

Consider the queueing systems $M/E_2/1$ and $M/H_2/1$ with service processes represented by $\mathbf{N}$ and $\mathbf{H}$, respectively. Since the matrices $\mathbf{S}_0$ and $\mathbf{S}_1$ have a Kronecker product form, the vector $\vec{\alpha}$ is itself the Kronecker product of the vectors $\vec{\alpha}_\mathbf{N}$ and $\vec{\alpha}_\mathbf{H}$, corresponding to the component PH distributions $\mathbf{N}$ and $\mathbf{H}$ of the KPC-PH process. Exploiting the Kronecker product form of $\vec{\alpha}$, we find that the Kronecker product of the solution matrices $\mathbf{G}_\mathbf{N}$ and $\mathbf{G}_\mathbf{H}$ of the decomposed

queues yields exactly the matrix $\mathbf{G}$ of the original $M/PH/1$ queue. That is,

$$\mathbf{G} = \mathbf{1}\tilde{\alpha} = (\mathbf{1}_n \otimes \mathbf{1}_h)(\tilde{\alpha}_N \otimes \tilde{\alpha}_H) = (\mathbf{1}_n\tilde{\alpha}_N) \otimes (\mathbf{1}_h\tilde{\alpha}_H) = \mathbf{G}_N \otimes \mathbf{G}_H$$

This example demonstrates that the matrix $\mathbf{G}$ of the $M/PH/1$ queue can be *exactly* factorised into the Kronecker product of the matrices $\mathbf{G}_N$ and $\mathbf{G}_H$ for the $M/E_2/1$ and $M/H_2/1$ queueing models, respectively. This property forms the basis for the efficient exact and approximate analysis techniques for QBD processes parameterized by KPC-PH service processes developed in the next section, which we later apply to fitting problems with general PH arrivals, where the above explicit solutions are not available.

5.3 Obtaining the queue-length distribution

In this section, we establish the theoretical results underlying the algorithm for efficiently computing the queue-length distribution for a QBD process with a given pair of arrival and service processes. Section 5.3.1 introduces the notation and formally defines component queues. Section 5.3.2, presents an efficient method to approximate the solution matrix $\mathbf{G}$, defined in the background chapter in (2.6), from the component queues. Section 5.3.3 uses the approximation of matrix $\mathbf{G}$ to estimate matrix $\mathbf{R}$ in (2.4), which is necessary for computing the queue-length probabilities. Finally, Section 5.3.4 offers a component queue-based approximation of the queue-length distribution.

5.3.1 Introducing component queues

Let $\mathbf{A}/S/1$ denote, analogously to Kendall's notation, a KPC-queue with a PH renewal arrival process $\mathbf{A}$ and KPC-PH service process $S = \mathbf{N} \otimes \mathbf{H}_1 \otimes \dots \otimes \mathbf{H}_J$. We follow the convention that scalar multiplication of a process applies to all the representation matrices, i.e., $c\mathbf{A} = (c\mathbf{A}_0, c\mathbf{A}_1)$ for real c . Our main result is a strategy to decompose the KPC-PH queue into a collection of smaller queueing systems, called *component queues*, enabling a divide-and-conquer solution strategy for the QBD model. We begin by formally defining component queues.

Definition 5.1 (Component queues). *Consider a KPC queue $A/S/1$ where $S = N \otimes H_1 \otimes \dots \otimes H_J$ is the Kronecker product composition of a general PH and J hyperexponential distributions. The following set of queueing processes gives its decomposition into component queues:*

1. $CQ_N := A/N^*/1$ where $N^* := \mu_{H_1}^* \dots \mu_{H_J}^* N$, and $\mu_{H_k}^*$ is the mean departure rate of process H_k , $k = 1, \dots, J$.
2. $CQ_{H_k} := M/H_k^*/1$, where M is a Poisson process with unit rate and

$$H_k^* := (\mu_N^*/\lambda_A^*) \left(\prod_{1 \leq i \leq J, i \neq k} \mu_{H_i}^* \right) H_k$$

where μ_N^* is the mean departure rate of process N , $\mu_{H_i}^*$ are the mean departure rates of processes H_i , and λ_A^* is the mean arrival rate of process A .

The component queues CQ_N and CQ_{H_k} , $1 \leq k \leq J$ defined above are themselves QBD processes. Moreover, each component queue has the same utilisation as the original $A/S/1$ queue. The latter property can be readily verified by the definitions, using the recursive expression of the Kronecker product composition moments in (2.2) for $k = 1$, as service rates are reciprocals of mean service times. Thus, if the $A/S/1$ queue is stable, so are its component queues.

5.3.2 Approximating the matrix G

We approximate the matrix G of the original queue via the matrices G_N and G_{H_k} , $k = 1, \dots, J$, which are analogously defined as G , but for the component queues CQ_N and CQ_{H_k} . Let $\tilde{G}$ denote the approximation of the matrix G . As demonstrated in the motivating example in Section 5.2, in the case of Poisson arrivals, $\tilde{G}$ is the same as G , so the approximation holds without error. Formally, we have:

Proposition 5.3.1. *Let $M/N \otimes H_1 \otimes \dots \otimes H_J/1$ be a KPC-PH queue with Poisson arrivals and let $\tilde{G} = G_N \otimes G_{H_1} \otimes \dots \otimes G_{H_J}$. Then $G = \tilde{G} = 1\bar{\alpha}$, where $\bar{\alpha}$ is the embedded distribution of the service process phase at departure instants.*

Proof. Let $\vec{\alpha}_S$ be the initial probability vector of the KPC-PH distribution and $\vec{\alpha}_N, \vec{\alpha}_{H_1}, \dots, \vec{\alpha}_{H_J}$ are that of the PH components of the KPC. For the original queue $M/S/1$ first, we have [LR99]

$$\mathbf{G} = \mathbf{1}_s \vec{\alpha}_S$$

Furthermore, for the component queues $CQ_N, CQ_{H_1}, \dots, CQ_{H_J}$ combined with the definition of $\tilde{\mathbf{G}}$, we get

$$\tilde{\mathbf{G}} = \mathbf{1}_n \vec{\alpha}_N \otimes \mathbf{1}_{h_1} \vec{\alpha}_{H_1} \otimes \dots \otimes \mathbf{1}_{h_J} \vec{\alpha}_{H_J}$$

Using the mixed-product property of the Kronecker product, i.e.,

$$AB \otimes CD = (A \otimes C)(B \otimes D)$$

the above equation can be rearranged to

$$\tilde{\mathbf{G}} = (\mathbf{1}_n \otimes \mathbf{1}_{h_1} \dots \otimes \mathbf{1}_{h_J})(\vec{\alpha}_N \otimes \vec{\alpha}_{H_1} \otimes \dots \otimes \vec{\alpha}_{H_J}) = \mathbf{1}_s \vec{\alpha}_S$$

where the last equation holds by the definition of a KPC-PH distribution. $\square$

While the previous section gives an exact decomposition of the $\mathbf{G}$ matrix for queues with Poisson inputs, if the arrival process is an arbitrary PH then $\mathbf{G}$ and $\tilde{\mathbf{G}}$ are no longer identical and our method provides an approximation. The following result outlines for this case the exact relationship between the original matrix $\mathbf{G}$ and its approximation $\tilde{\mathbf{G}}$.

Theorem 5.1. *Consider a queue with PH arrival process $\mathbf{A}$ and KPC-PH service process $\mathbf{S} = \mathbf{N} \otimes \mathbf{H}_1 \otimes \dots \otimes \mathbf{H}_J$, where $\mathbf{H}_k, k = 1, \dots, J$ are hyperexponential. Defining $\tilde{\mathbf{G}} = \mathbf{G}_N \otimes \mathbf{G}_{H_1} \otimes \dots \otimes \mathbf{G}_{H_J}$, we have the equation*

$$\tilde{\mathbf{G}} = (\mathbf{I}_{an} \otimes \mathbf{G}_{H_1} \otimes \dots \otimes \mathbf{G}_{H_J}) \mathbf{G} \quad (5.1)$$

where a and n denote the number of phases of process $\mathbf{A}$ and $\mathbf{N}$, respectively, and $\mathbf{I}_{an}$ is the identity matrix with dimension $an \times an$.

Before presenting the proof, we give the following definition.

Definition 5.2. *The state space S of a QBD process with arrival process A and service process $S = N \otimes H_1 \otimes \dots \otimes H_J$ can be described with the tuple $S = (A, N, H_1, \dots, H_J, L)$ where $A \in \{1, \dots, a\}$, $N \in \{1, \dots, n\}$ and $H_k \in \{1, \dots, h_k\}$ denote the internal phases of the arrival and service processes and $L \in \mathbb{N}_0$ denotes the level of the process or in other words the queue length.*

Proof. To simplify the notation used throughout the proof, we observe the following:

1. Using Proposition 5.3.1, we know that for Poisson arrivals the approximation is exact. Here, each component queue, except the first one, has Poisson arrivals. This implies that if we denote $H = H_1 \otimes \dots \otimes H_J$ and consider a new combined component queue CQ_H with Poisson arrivals and service H , that is $M/H/1$, then $G_H = G_{H_1} \otimes \dots \otimes G_{H_J}$. Therefore, without loss of generality, we can assume that the number of hyperexponential components in the KPC service is $J = 1$, so that $S = N \otimes H$.
2. As the QBD process in question is homogeneous (2.3), G does not depend on the level of the process. Henceforth, we assume that in (2.7), the starting level is $L = 1$ and the stopping level is $L = 0$, omitting the last coordinate from the state space notation given in Definition 5.2 for brevity.

We begin with examining the elements of matrix $\tilde{G}$. By definition $\tilde{G} = G_N \otimes G_H$. Starting with G_N and the corresponding component queue $CQ_N = A/N^*/1$, denote the QBD process as $Y_t \in (A, N, L)$ for $t \geq 0$. Furthermore, denote the elements of G_N as g_{ij}^N where $i, j \in (A, N)$. Using (2.7), given a state (x, y) in level 1 and a state (x', y') in level 0,

$$g_{(x,y),(x',y')}^N = P(t_0 < \infty \wedge Y_{t_0} = (x', y') | Y_0 = (x, y))$$

where t_0 is the first passage time corresponding to reaching level 0, given the process started in level 1. We introduce the following notation for brevity. Let

- s_0 denote the first state the process enters when reaching level 0 at $t_0 < \infty$

- s_1 denote the state of level 1 from where the last transition before reaching level 0 occurs

Conditioning on the value s_1 takes, we get

$$\begin{aligned} g_{(x,y),(x',y')}^N &= P(s_0 = (x', y') | Y_0 = (x, y)) \\ &= \sum_{(x_k, y_k) \in (A, N)} P(s_0 = (x', y') | s_1 = (x_k, y_k), Y_0 = (x, y)) P(s_1 = (x_k, y_k) | Y_0 = (x, y)) \end{aligned}$$

Downward transitions are only possible if $x_k = x'$ because, upon a departure, the phase of the *arrival* process cannot change. Furthermore, since N^* is a PH process, the probability of entering level 0 at a specific phase does not depend on the phase from which the process departed in level 1. Therefore, we have

$$\forall y_k \in N : \quad P(s_0 = (x', y') | s_1 = (x_k, y_k), Y_0 = (x, y)) = \begin{cases} \tilde{\alpha}_N(y') & \text{if } x_k = x' \\ 0 & \text{otherwise} \end{cases}$$

So that,

$$g_{(x,y),(x',y')}^N = \tilde{\alpha}_N(y') \sum_{y_k \in N} P(s_1 = (x', y_k) | Y_0 = (x, y)) \quad (5.2)$$

Due to Proposition 5.3.1, we also know that for the elements of G_H

$$g_{z,z'}^H = \tilde{\alpha}_H(z') \quad (5.3)$$

Using the definition of the Kronecker product, the elements of $\tilde{G} = G_H \otimes G_N$ can now be expressed by taking the product of (5.2) and (5.3).

$$\begin{aligned} \tilde{g}_{(x,y,z),(x',y',z')} &= g_{(x,y),(x',y')}^N g_{z,z'}^H \\ &= \alpha_N(y') \alpha_H(z') \sum_{y_k \in N} P(s_1 = (x', y_k) | Y_0 = (x, y)) \\ &= \alpha_S((y', z')) \sum_{y_k \in N} P(s_1 = (x', y_k) | Y_0 = (x, y)) \end{aligned} \quad (5.4)$$

The goal is to prove (5.1). We restate the equation here for $J = 1$ for reading convenience.

$$\tilde{G} = (I_{an} \otimes G_H) G$$

Let us now focus on the right-hand side of the equation. Similar to how we derived the expression for the elements of $\mathbf{G}_N$ in (5.2), the elements of $\mathbf{G}$ can be expressed as follows. Let X_t denote the stochastic process where $X_t \in (A, N, H, L)$ for $t \geq 0$ and let g_{ij} denote the elements of $\mathbf{G}$. Starting from state (x, y, z) of level 1 and stopping in state (x', y', z') of level 0, we get

$$g_{(x,y,z),(x',y',z')} = \alpha_S((y', z')) \sum_{y_k \in N} \sum_{z_k \in H} P(s_1 = (x', y_k, z_k) | X_0 = (x, y, z)) \quad (5.5)$$

Note that multiplying with $(\mathbf{I}_{an} \otimes \mathbf{G}_H)$, is effectively a deconditioning of the probabilities $g_{(x,y,z),(x',y',z')}$ with respect to the third coordinate of the starting state — specifically, the starting phase of process $\mathbf{H}$, that is

$$\begin{aligned} & \sum_{z \in H} \alpha_H(z) g_{(x,y,z),(x',y',z')} \\ &= \sum_{z \in H} \alpha_H(z) \left(\alpha_S((y', z')) \sum_{y_k \in N} \sum_{z_k \in H} P(s_1 = (x', y_k, z_k) | X_0 = (x, y, z)) \right) \\ &= \alpha_S((y', z')) \sum_{y_k \in N} \sum_{z_k \in H} P(s_1 = (x', y_k, z_k) | X_0 \in \Sigma) \end{aligned} \quad (5.6)$$

where $\Sigma = \{(x, y, 1), \dots, (x, y, h)\}$ represents the set of states with first two coordinates x and y , and an arbitrary third coordinate corresponding to the phase of process $\mathbf{H}$. Our final task is to show that the expression for the elements of $(\mathbf{I}_{an} \otimes \mathbf{G}_H)\mathbf{G}$ in (5.6) is equal to the expression we have for the elements of $\tilde{\mathbf{G}}$ in (5.4). To show this, note that the component queue $\mathbf{A}/N^*/1$ corresponding to Y_t is an aggregated version of the original stochastic process X_t . Comparing $\mathbf{A}/S/1$ to $\mathbf{A}/N^*/1$, the states and transition rates change as follows:

1. The states $\{(x, y, 1, l), \dots, (x, y, h, l)\}$ of X_t are merged into one state (x, y, l) of process Y_t , since Y_t ignores the phases of $\mathbf{H}$.
2. The transition rates corresponding to departures in the original queue are replaced with weighted averages in the component queue. Let $\mu_{y,y'}\gamma_{z_k}$ be the transition rate from state (x, y, z) of level 1 to state (x, y', z_k) of level 0 and note that $\mu_{y,y'}\gamma_{z_k}$ does not depend on z . In the aggregated system, this is a transition from state (x, y) of level 1 to state (x, y') of level 0 with rate

$$\mu_{y,y'}^* = \mu_{y,y'} \sum_{z_k \in H} \gamma_{z_k} \pi_H(z_k) = \mu_{y,y'} \mu_H^*$$

where $\vec{\pi}_H = (\pi_H(1), \dots, \pi_H(h))$ is the steady-state probability vector for phases $\{1, \dots, h\}$ within process H and μ_H^* is the mean departure rate of H , as specified in Definition 5.1. This implies that the transition rate from a set of states $\{(x, y, 1), \dots, (x, y, h)\}$ to another set $\{(x', y', 1), \dots, (x', y', h)\}$ in the original system is the same as the transition rate from state (x, y) to state (x', y') in the aggregated system.

Consequently,

$$\begin{aligned} \sum_{z_k \in H} P(s_1 = (x', y_k, z_k) | X_0 \in \{(x, y, 1), \dots, (x, y, h)\}) = \\ P(s_1 \in \{(x', y_k, 1), \dots, (x', y_k, h)\} | X_0 \in \{(x, y, 1), \dots, (x, y, h)\}) = P(s_1 = (x', y_k) | Y_0 = (x, y)) \end{aligned} \quad (5.7)$$

Using (5.7) to transform (5.6), we obtain

$$\sum_{z \in H} \tilde{\alpha}_H(z) g_{(x, y, z), (x', y', z')} = \alpha_S((y', z')) \sum_{y_k \in N} P(s_1 = (x', y_k) | Y_0 = (x, y)) \quad (5.8)$$

which is the same expression we have derived for $\tilde{G}$ in (5.4). This concludes the proof that the weighted average of the elements of G is identical to the elements of $\tilde{G}$. $\square$

As we explain in the following sections, the above result is fundamental as it allows us to estimate steady-state performance metrics for the QBD process in terms of the G_N and G_{H_k} matrices of the individual component queues CQ_N and CQ_{H_k} . As these processes are significantly smaller than the original queue, the net result is a large reduction in the computational cost of the QBD analysis.

The key result in Theorem 5.1 is interpreted as follows. Matrix $\tilde{G} = G_N \otimes G_{H_1} \otimes \dots \otimes G_{H_J}$, which is efficiently constructed with a Kronecker product of matrices each obtained by solving a smaller component queue, may be seen as an approximation of G by means of aggregation. The aggregation is applied by the pre-multiplying factors on the right-hand side of (5.1). Compared to the exact model, relying on $\tilde{G}$ thus loses some information about the internal phases of process H , but its computation can be done much more efficiently than computing G . This

provides us with a novel approximation for the solution of the QBD, which in the case of Poisson arrivals also becomes exact.

Example. To illustrate (5.1) in concrete terms, we now show it in a small example. Let the number of hyperexponential components in the KPC service be $J = 1$, and the degree of the arrival process $\mathbf{A}$ and service components $\mathbf{N}, \mathbf{H}$ be 2, so that $an = 4$ and let $\alpha_H = [\alpha_1, \alpha_2]$ be the embedded distribution of the hyperexponential component. From Proposition 5.3.1, we know that $\mathbf{G}_H = \mathbf{1}\bar{\alpha}_H$, so that

$$\mathbf{I}_4 \otimes \mathbf{G}_H = \begin{bmatrix} \alpha_1 & \alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha_1 & \alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_1 & \alpha_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_1 & \alpha_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 \end{bmatrix}$$

Multiplying $\mathbf{I}_4 \otimes \mathbf{G}_H$ with $\mathbf{G} = (g_{ij})_{1 \leq i, j, \leq m}$ implies that the $(2i - 1)$ th and $2i$ th rows of $\tilde{\mathbf{G}}$ are identical. Furthermore, they are computed from $\mathbf{G}$ by averaging these specific rows, weighted according to the embedded distribution $\bar{\alpha}_H$ of $\mathbf{H}$. That is,

$$\tilde{\mathbf{G}} = \mathbf{G}_N \otimes \mathbf{G}_H = (\mathbf{I}_4 \otimes \mathbf{G}_H)\mathbf{G} = \begin{bmatrix} \alpha_1 g_{11} + \alpha_2 g_{21} & \alpha_1 g_{12} + \alpha_2 g_{22} & \dots & \alpha_1 g_{18} + \alpha_2 g_{28} \\ \alpha_1 g_{11} + \alpha_2 g_{21} & \alpha_1 g_{12} + \alpha_2 g_{22} & \dots & \alpha_1 g_{18} + \alpha_2 g_{28} \\ \alpha_1 g_{31} + \alpha_2 g_{41} & \alpha_1 g_{32} + \alpha_2 g_{42} & \dots & \alpha_1 g_{38} + \alpha_2 g_{48} \\ \alpha_1 g_{31} + \alpha_2 g_{41} & \alpha_1 g_{32} + \alpha_2 g_{42} & \dots & \alpha_1 g_{38} + \alpha_2 g_{48} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1 g_{71} + \alpha_2 g_{81} & \alpha_1 g_{72} + \alpha_2 g_{82} & \dots & \alpha_1 g_{78} + \alpha_2 g_{88} \end{bmatrix}$$

which shows the aforementioned aggregation property. We also provide a numerical example to illustrate the aggregation process further. Consider a queue with the following hyperexponential

arrival process with mean 3 and squared coefficient of variation (SCV) 11/9,

$$\mathbf{A}_0 = \begin{bmatrix} -0.25 & 0 \\ 0 & -0.5 \end{bmatrix} \quad \mathbf{A}_1 = \begin{bmatrix} 0.125 & 0.125 \\ 0.25 & 0.25 \end{bmatrix}$$

and a KPC service process where the first component is Erlang-2 with unit mean and the second component is a 2-phase hyperexponential distribution with the following parameters

$$\mathbf{H}_0 = \begin{bmatrix} -10 & 0 \\ 0 & -20 \end{bmatrix} \quad \mathbf{H}_1 = \begin{bmatrix} 5 & 5 \\ 5 & 15 \end{bmatrix}$$

corresponding to mean 1/15 and SCV 5/4. Below we can see how closely $\mathbf{G}$ aligns with the aggregated matrix $\tilde{\mathbf{G}} = \mathbf{G}_N \otimes \mathbf{G}_H$,

$$\mathbf{G} = \begin{bmatrix} 0.488 & 0.488 & 0 & 0 & 0.012 & 0.012 & 0 & 0 \\ 0.494 & 0.494 & 0 & 0 & 0.006 & 0.006 & 0 & 0 \\ 0.494 & 0.494 & 0 & 0 & 0.006 & 0.006 & 0 & 0 \\ 0.497 & 0.497 & 0 & 0 & 0.003 & 0.003 & 0 & 0 \\ 0.024 & 0.024 & 0 & 0 & 0.476 & 0.476 & 0 & 0 \\ 0.012 & 0.012 & 0 & 0 & 0.488 & 0.488 & 0 & 0 \\ 0.012 & 0.012 & 0 & 0 & 0.488 & 0.488 & 0 & 0 \\ 0.006 & 0.006 & 0 & 0 & 0.494 & 0.494 & 0 & 0 \end{bmatrix} \quad \tilde{\mathbf{G}} = \begin{bmatrix} 0.491 & 0.491 & 0 & 0 & 0.009 & 0.009 & 0 & 0 \\ 0.491 & 0.491 & 0 & 0 & 0.009 & 0.009 & 0 & 0 \\ 0.496 & 0.496 & 0 & 0 & 0.005 & 0.005 & 0 & 0 \\ 0.496 & 0.496 & 0 & 0 & 0.005 & 0.005 & 0 & 0 \\ 0.018 & 0.018 & 0 & 0 & 0.482 & 0.482 & 0 & 0 \\ 0.018 & 0.018 & 0 & 0 & 0.482 & 0.482 & 0 & 0 \\ 0.009 & 0.009 & 0 & 0 & 0.491 & 0.491 & 0 & 0 \\ 0.009 & 0.009 & 0 & 0 & 0.491 & 0.491 & 0 & 0 \end{bmatrix}$$

As visible, the proposed approximation based on component queues effectively provides a local weighted sum of neighbouring elements of $\mathbf{G}$. The benefit is that the expression on the right can be computed efficiently via Kronecker products, whereas the original $\mathbf{G}$ matrix on the left cannot.

5.3.3 Approximating the matrix $\mathbf{R}$

The formula (2.4), which is used for calculating the steady-state queue-length probability vectors, requires the calculation of matrix $\mathbf{R}$. In this subsection, $\tilde{\mathbf{G}}$, derived previously, is used to obtain an estimate for $\mathbf{R}$. Using (2.5) and substituting the approximation $\tilde{\mathbf{G}}$ for $\mathbf{G}$, we propose

the approximation

$$R \approx -F(L + F\tilde{G})^{-1} \quad (5.9)$$

In the above formula, the computational bottleneck is the inverse calculation, which however we now show has special properties for the class of QBD we consider. First, we have found through numerical experimentation that the inversion of matrix L is very efficient due to its sparsity. Furthermore, we prove in the following result that $F\tilde{G}$ is a low-rank matrix.

Theorem 5.2. *Given a KPC queue with PH arrival process $A = (A_0, A_1)$ and service process $S = N \otimes H_1 \otimes \dots \otimes H_J$, where $H_k, k = 1, \dots, J$ are hyperexponential, and $\tilde{G} = G_N \otimes G_{H_1} \otimes \dots \otimes G_{H_J}$ as before, $\text{rank}(F\tilde{G}) \leq n$, where n is the number of phases of process N .*

Proof. F records the forward transitions of the QBD process. Its definition is given by (2.3). In the case of the component queues, using the definition, we have $F_N = A_1 \otimes I_n$ and $F_k = I_{h_k}, k = 1, \dots, J$. Therefore,

$$\begin{aligned} F &= A_1 \otimes I_s = A_1 \otimes (I_n \otimes I_{h_1} \otimes \dots \otimes I_{h_n}) = (A_1 \otimes I_n) \otimes I_{h_1} \otimes \dots \otimes I_{h_n} \\ &= F_N \otimes F_{H_1} \otimes \dots \otimes F_{H_n} \end{aligned}$$

Using the above together with the definition of $\tilde{G}$, the product $F\tilde{G}$ can be rearranged as follows

$$\begin{aligned} F\tilde{G} &= (F_N \otimes F_{H_1} \otimes \dots \otimes F_{H_n}) (G_N \otimes G_{H_1} \otimes \dots \otimes G_{H_J}) \\ &= (F_N G_N) \otimes (F_{H_1} G_{H_1}) \otimes \dots \otimes (G_{H_J} F_{H_n}) \end{aligned}$$

Next, we note two well-known results for the ranks of (Kronecker) product form matrices [LS04].

$$\text{rank}(A \otimes B) = \text{rank}(A)\text{rank}(B) \quad (5.10)$$

$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)) \quad (5.11)$$

Applying (5.10), we get

$$\text{rank}(F\tilde{G}) = \text{rank}(F_N G_N) \text{rank}(F_{H_1} G_{H_1}) \dots \text{rank}(F_{H_J} G_{H_J}) \quad (5.12)$$

As seen in Proposition 5.3.1, $\text{rank}(G_{H_k}) = 1, k = 1, \dots, J$. So that using (5.11) above, we have

$$\text{rank}(\mathbf{F}_{H_k} \mathbf{G}_{H_k}) \leq \min(\text{rank}(\mathbf{F}_{H_k}), 1) = 1 \quad k = 1, \dots, J \quad (5.13)$$

Furthermore, we know that $\mathbf{F}_N = \mathbf{A}_1 \otimes \mathbf{I}_n$. Since the arrival process $\mathbf{A}$ is PH, $\text{rank}(\mathbf{A}_1) = 1$.

Using (5.11) again, we get

$$\text{rank}(\mathbf{F}_N) = \text{rank}(\mathbf{A}_1 \otimes \mathbf{I}_n) = n$$

Consequently,

$$\text{rank}(\mathbf{F}_N \mathbf{G}_N) \leq \min(n, \text{rank}(\mathbf{G}_N)) \leq n \quad (5.14)$$

Finally, substituting (5.13) and (5.14) into (5.12), we get

$$\text{rank}(\mathbf{F} \tilde{\mathbf{G}}) \leq n$$

which concludes the proof. $\square$

The low rank of $\mathbf{F} \tilde{\mathbf{G}}$ is an important property of our method, which allows us to efficiently compute the inverse of the sum in (5.9) by utilising the Woodbury matrix identity,

$$\mathbf{R} \approx -\mathbf{F}(\mathbf{L} + \mathbf{F} \tilde{\mathbf{G}})^{-1} = -\mathbf{F}(\mathbf{L} + \mathbf{U} \Sigma \mathbf{V})^{-1} = -\mathbf{F}(\mathbf{L}^{-1} - \mathbf{L}^{-1} \mathbf{U} (\Sigma^{-1} + \mathbf{V} \mathbf{L}^{-1} \mathbf{U})^{-1} \mathbf{V} \mathbf{L}^{-1}) \quad (5.15)$$

where $\mathbf{F}, \mathbf{L}, \tilde{\mathbf{G}} \in \mathbb{R}^{m \times m}$, $\mathbf{U} \in \mathbb{R}^{m \times k}$, $\Sigma \in \mathbb{R}^{k \times k}$, $\mathbf{V} \in \mathbb{R}^{k \times m}$, and $k = \text{rank}(\mathbf{F} \tilde{\mathbf{G}})$. The Woodbury matrix identity relies on the decomposition of $\mathbf{F} \tilde{\mathbf{G}} = \mathbf{U} \Sigma \mathbf{V}$, where Σ is invertible and hence $\text{rank}(\Sigma) = \text{rank}(\mathbf{F} \tilde{\mathbf{G}})$. A straightforward approach to obtain $\mathbf{U}, \Sigma, \mathbf{V}$ is to use the Singular Value Decomposition (SVD) of $\mathbf{F} \tilde{\mathbf{G}}$. The computational complexity of this operation for a matrix with dimension $m \times m$ is $\mathcal{O}(m^3)$ in general [BR05]. However, we can improve that by exploiting the special Kronecker-form structure of $\mathbf{F} \tilde{\mathbf{G}}$. We know that $\mathbf{F}_{H_k} \mathbf{G}_{H_k} = \mathbf{I}_{h_k} (\mathbf{1}_{h_k} \tilde{\alpha}_{H_k}) = \mathbf{1}_{h_k} \tilde{\alpha}_{H_k}$, $k = 1, \dots, J$ from Proposition 5.3.1. Let $\mathbf{F}_N \mathbf{G}_N = \mathbf{U}_N \Sigma \mathbf{V}_N$ be the SVD of matrix $\mathbf{F}_N \mathbf{G}_N$. Define $\mathbf{U}$ and $\mathbf{V}$ as follows

$$\begin{aligned} \mathbf{F} \tilde{\mathbf{G}} &= \mathbf{F}_N \mathbf{G}_N \otimes \mathbf{F}_{H_1} \mathbf{G}_{H_1} \otimes \dots \otimes \mathbf{F}_{H_J} \mathbf{G}_{H_J} = (\mathbf{U}_N \Sigma \mathbf{V}_N) \otimes (\mathbf{1}_{h_1} \tilde{\alpha}_{H_1}) \otimes \dots \otimes (\mathbf{1}_{h_J} \tilde{\alpha}_{H_J}) \\ &= (\mathbf{U}_N \otimes \mathbf{1}_{h_1} \otimes \dots \otimes \mathbf{1}_{h_J}) \Sigma (\mathbf{V}_N \otimes \tilde{\alpha}_{H_1} \dots \otimes \alpha_{H_J}) = \mathbf{U} \Sigma \mathbf{V} \end{aligned} \quad (5.16)$$

Since $\text{rank}(\mathbf{F}\tilde{\mathbf{G}}) = \text{rank}(\mathbf{F}_N\mathbf{G}_N) = \text{rank}(\Sigma)$, this decomposition has the required properties. Moreover, it is significantly faster to calculate as it only requires the SVD of a much smaller matrix $\mathbf{F}_N\mathbf{G}_N$. The SVD of $\mathbf{F}_N\mathbf{G}_N$ takes $\mathcal{O}(n^3)$ time and calculating the Kronecker products to obtain $\mathbf{U}$ and $\mathbf{V}$ takes $\mathcal{O}(m^2)$ time. Therefore, the entire decomposition runs in $\mathcal{O}(n^3 + m^2)$ time. Recall that the size of $\mathbf{F}_N\mathbf{G}_N$ is $n \times n$ and $\mathbf{F}\tilde{\mathbf{G}}$ is a $m \times m$ matrix. Under the assumption that $n \ll m$, $\mathcal{O}(n^3 + m^2)$ is dominated by the m^2 factor.

5.3.4 Approximating the queue-length probabilities

With the efficient approximation of matrix $\mathbf{R}$ in (2.4), we can now compute the steady-state probability vectors $\tilde{\pi}_l = (\pi_{l1}, \dots, \pi_{lm})$ for $l \in \mathbb{N}_0$, where m is the number of phases in each level. As stated in the background section, to calculate $\tilde{\pi}_0$, one must first solve the associated system of equations given by (2.8) and (2.9).

However, we now introduce a novel characterization result for the PH/PH/1 queue. A similar result is established for the MAP/G/1 queue in [BB06, Theorem 12.10]; unlike this earlier result, our characterization applies to situations where the service phase is tracked explicitly in the definition of the state space. We start by proving two auxiliary lemmas.

Lemma 5.3.1. *Let $\mathbf{G}' := \int_0^\infty e^{(\mathbf{L}_0 + \mathbf{F}\mathbf{G})t} dH(t)$ where $H(t)$ is the service time probability distribution. If $\vec{x} \in \mathbb{R}^{1 \times as}$ is a probability vector such that $\vec{x}\mathbf{G} = \vec{x}$, then $\vec{x}\mathbf{G}' = \vec{x}$.*

Proof. Given $\vec{x}\mathbf{G} = \vec{x}$, we must show that $\vec{x}\mathbf{G}' = \vec{x}$ also holds. Let $\mathbf{P} = \mathbf{I}_a \otimes \vec{\alpha}_s$ and $\mathbf{Q} = \mathbf{I}_a \otimes \mathbf{1}_s$ where $\alpha_s \in \mathbb{R}^{1 \times s}$ is the equilibrium probability vector of the embedded Markov chain at departure instants and $\mathbf{1}_s \in \mathbb{R}^{s \times 1}$ is a column vector of ones. Due to the probabilistic interpretation of $\mathbf{G}$, we have

$$\mathbf{G}\mathbf{Q}\mathbf{P} = \mathbf{G} \tag{5.17}$$

This is because multiplying with $\mathbf{Q}$ sums the columns of $\mathbf{G}$ belonging to each arrival phase, thus calculating the aggregated probability of entering a given arrival phase on level $l - 1$, whereas multiplication with $\mathbf{P}$ reverses this aggregation by expanding the aggregated probabilities in proportion to $\vec{\alpha}$. This returns the original probabilities in $\mathbf{G}$, since the service process is PH and the last transition is a departure during which the service phase is generated with probability

$\vec{\alpha}_S$ irrespective of the state the last transition started from. We also have that

$$\vec{x} = \vec{x}G = \vec{x}GQP = \vec{x}QP \quad (5.18)$$

Now, introducing the notation $\vec{y} = \vec{x}Q$, (5.18) becomes $\vec{x} = \vec{y}P$ and $\vec{x} = \vec{x}G$ can be rewritten as $\vec{y}PG = \vec{y}P$. Multiplying both sides by Q and noting that $PQ = I$, we get

$$\vec{y}PGQ = \vec{y}PQ = \vec{y} \quad (5.19)$$

Let $G_A = PGQ \in \mathbb{R}^{a \times a}$. Since $\vec{y}\mathbf{1} = 1$, $\vec{y}$ is the stationary probability vector of G_A . By definition, G_A is an aggregation of G and its elements are the aggregated probabilities corresponding to starting at a given arrival phase of level l and being in a specific arrival phase when entering level $l - 1$ for the first time. This aggregation treats the original $PH/PH/1$ queue as a $PH/G/1$ queue. Therefore, using [BB06, Theorem 12.8], G_A can also be calculated as

$$G_A = \int_0^\infty e^{(A_0 + A_1 G_A)t} dH(t) \quad (5.20)$$

Next, we show that by performing this aggregation on the DTMC governed by G' we get probabilities equal to G_A . To see this, we calculate the infinitesimal generator of the aggregated system from $(L_0 + FG)$, which itself defines an infinitesimal generator.

$$P(L_0 + FG)Q = PL_0Q + PFGQ$$

Using the definition of the matrices and the mixed-product property of the Kronecker product $(C_1 \otimes C_2)(C_3 \otimes C_4) = (C_1C_3 \otimes C_2C_4)$, we have

$$\begin{aligned} PL_0Q &= (I_a \otimes \vec{\alpha}_S)(A_0 \otimes I_S)(I_a \otimes \mathbf{1}_s) = A_0 \\ PFGQ &= (I_a \otimes \vec{\alpha}_S)(A_1 \otimes I_S)GQ = (A_1 \otimes \vec{\alpha}_S)GQ = \\ &= (I_a \otimes \vec{\alpha}_S)(A_1 \otimes I_S)(I_a \otimes \mathbf{1}_s)(I_a \otimes \vec{\alpha}_S)GQ = A_1(I_a \otimes \vec{\alpha}_S)GQ = \\ &= A_1PGQ = A_1G_A \end{aligned}$$

Hence, the infinitesimal generator becomes $A_0 + A_1 G_A$ and using (5.20), we get $PG'Q = G_A$. Since $\vec{y}$ is a stationary vector of G_A , we have $\vec{y}PG'Q = \vec{y}$ and multiplying with P both sides we get that $\vec{x} = \vec{y}P$ is the stationary probability vector of G' as stated. $\square$

Lemma 5.3.2. *If $\vec{x} \in \mathbb{R}^{1 \times as}$ such that $\vec{x} = \vec{x}G'$ and $\vec{x}\mathbf{1} = 1$ then $\vec{x}(L_0 + FG) = 0$ and $\vec{x}$ is the stationary distribution corresponding to the original process observed only during idle periods.*

Proof. Since $L_0 + FG$ describes the phase transitions of the original CTMC when observed only during idle periods, it is irreducible given the original CTMC is irreducible. Furthermore, the DTMC governed by G' is also irreducible since it describes a time-averaged behaviour of this truncated CTMC. Hence, both Markov chains have a unique stationary distribution. Let $\vec{y}$ be the stationary vector of $L_0 + FG$, that is $\vec{y}(L_0 + FG) = 0$. Expanding the matrix exponential operation as a series we get

$$\begin{aligned} \vec{y}G' &= \vec{y} \int_0^\infty \sum_{k=0}^\infty \frac{(L_0 + FG)^k}{k!} t^k dH(t) \\ &= \vec{y} + \int_0^\infty \sum_{k=1}^\infty \frac{\vec{y}(L_0 + FG)^k}{k!} t^k dH(t) = \vec{y} \end{aligned}$$

Therefore, $\vec{y}$ is also the stationary probability vector of G' and due to the uniqueness of stationary probabilities, we must have $\vec{y} = \vec{x}$. $\square$

Theorem 5.3. *Let $\vec{\pi}_0 = (\pi_{01}, \dots, \pi_{0m})$, then*

$$\vec{\pi}_0 = (1 - \rho)\vec{x} \tag{5.21}$$

where $\vec{x}$ is the solution to $\vec{x}G = \vec{x}$, $\vec{x}\mathbf{1}_m = 1$.

Proof. From Lemma 5.3.1 we have that $\vec{x}G' = \vec{x}$ and using Lemma 5.3.2 we know that $\vec{x}$ is the stationary distribution of the CTMC when observed only during idle periods. In other words, $\vec{x}$ corresponds to the conditional steady-state probabilities of the original process given that it is at level 0. The probability that the queue is empty is $1 - \rho$. Thus, after deconditioning, we get $\vec{\pi}_0 = \vec{x}(1 - \rho)$ as stated. $\square$

It is known that if a solution $\vec{x}$ exists, then it is unique for the system. Using our approximate $\tilde{\mathbf{G}} = \mathbf{G}_N \otimes \mathbf{G}_{H_1} \otimes \cdots \otimes \mathbf{G}_{H_J}$, similarly to (5.21), we have that $\vec{x}\tilde{\mathbf{G}} = \vec{x}$, $\vec{x}\mathbf{1}_s = 1$. Let $\vec{x}_N$ and $\vec{x}_{H_k}$, $k = 1, \dots, J$ be the solutions of the above system of equation where $\tilde{\mathbf{G}}$ is replaced by $\mathbf{G}_N$ and $\mathbf{G}_{H_k}$, $k = 1, \dots, J$, respectively.

$$\begin{aligned} (\vec{x}_N \otimes \vec{x}_{H_1} \otimes \cdots \otimes \vec{x}_{H_J})\tilde{\mathbf{G}} &= (\vec{x}_N \otimes \vec{x}_{H_1} \otimes \cdots \otimes \vec{x}_{H_J})(\mathbf{G}_N \otimes \mathbf{G}_{H_1} \otimes \cdots \otimes \mathbf{G}_{H_J}) \\ &= \vec{x}_N \mathbf{G}_N \otimes \vec{x}_{H_1} \mathbf{G}_{H_1} \otimes \cdots \otimes \vec{x}_{H_J} \mathbf{G}_{H_J} \\ &= \vec{x}_N \otimes \vec{x}_{H_1} \otimes \cdots \otimes \vec{x}_{H_J} \end{aligned}$$

where the last equation holds because of the definition of the vectors $\vec{x}_i$. Furthermore,

$$\begin{aligned} (\vec{x}_N \otimes \vec{x}_{H_1} \otimes \cdots \otimes \vec{x}_{H_J})\mathbf{1}_s &= (\vec{x}_N \otimes \vec{x}_{H_1} \otimes \cdots \otimes \vec{x}_{H_J})(\mathbf{1}_n \otimes \mathbf{1}_{h_1} \otimes \cdots \otimes \mathbf{1}_{h_J}) \\ &= \vec{x}_N \mathbf{1}_n \otimes \vec{x}_{H_1} \mathbf{1}_{h_1} \otimes \cdots \otimes \vec{x}_{H_J} \mathbf{1}_{h_J} \\ &= 1 \otimes 1 \otimes \cdots \otimes 1 = 1 \end{aligned}$$

Therefore, $\vec{x}_N \otimes \vec{x}_{H_1} \otimes \cdots \otimes \vec{x}_{H_k}$ is the unique solution to the approximate system of equations $\vec{x}\tilde{\mathbf{G}} = \vec{x}$, $\vec{x}\mathbf{1}_s = 1$. Furthermore, Proposition 5.3.1 states that $\mathbf{G}_{H_k} = \mathbf{1}_{h_k} \vec{\alpha}_{H_k}$. This implies that $\vec{x}_{H_k} = \vec{\alpha}_{H_k}$, since $\vec{\alpha}_{H_k}$ is a probability vector and $\vec{\alpha}_{H_k} \mathbf{G}_{H_k} = \vec{\alpha}_{H_k} \mathbf{1}_{h_k} \vec{\alpha}_{H_k} = 1 \vec{\alpha}_{H_k} = \vec{\alpha}_{H_k}$.

Let $\tilde{\mathbf{R}}$ be the approximation of $\mathbf{R}$ given by (5.15). Based on the previous results, $\tilde{\pi}_l$ can be computed as

$$\begin{aligned} \tilde{\pi}_0 &= (1 - \rho)(\vec{x}_N \otimes \vec{\alpha}_{H_1} \otimes \cdots \otimes \vec{\alpha}_{H_J}) \\ \tilde{\pi}_l &= \tilde{\pi}_0 \tilde{\mathbf{R}}^l \quad \text{for } l > 0 \end{aligned}$$

5.4 Fitting algorithm

In this section, the theoretical results established in the previous section are combined with a search algorithm to find a QBD process that matches a given set of queue-length probabilities. Our main contribution is the optimisation problem formulation. The user may choose different

numerical solvers based on their preferred cost and convergence properties. Section 5.4.1 introduces the decision variables of the optimisation problem. Section 5.4.2 outlines the optimisation problem itself. Section 5.4.3 presents a method to reduce the dimension of the search problem by utilising gamma distributions. Section 5.4.4 shows how the fitted distribution is reduced to handle potential overfitting and overparameterization. Finally, a time and space complexity analysis is provided in Section 5.4.5 and details of the solver used for the optimisation problem are presented in Section 5.4.6.

5.4.1 Decision variables

The decision variables are a set of vector parameters used to construct the components of the KPC-PH process. Initially, we describe the construction of the hyperexponential PH components, denoted as $\mathbf{H}_i = (\mathbf{H}_{i0}, \mathbf{H}_{i1})$, $i = 1, \dots, J$, specified by the parameter vectors $\vec{\alpha}_{H_i} \in \mathbb{R}^{h_i-1}$ and $\vec{\lambda}_{H_i} \in \mathbb{R}^{h_i}$. Note, that $\vec{\alpha}_{H_i}$ has one dimension less than the number of phases in process $\mathbf{H}$, with the last element chosen to satisfy the constraint that the elements sum to 1. Therefore, the actual probability vector is formulated as $\vec{\alpha}'_{H_i} = (\vec{\alpha}_{H_i}, 1 - \vec{\alpha}_{H_i} \mathbf{1}_{h_i-1})$.

To ensure $\vec{\alpha}'_{H_i}$ is a valid probability vector, the following constraints apply to the decision variables: $\mathbf{0} \leq \vec{\alpha}_{H_i}$ and $\vec{\alpha}_{H_i} \mathbf{1} \leq 1$ for $i = 1, \dots, J$. The diagonal elements of $\mathbf{H}_{i0}$ are stored in the decision vector $\mathbf{0} \leq \vec{\lambda}_{H_i}$, $i = 1, \dots, J$, so that $\mathbf{H}_{i0} = \text{diag}(-\vec{\lambda}_{H_i})$ and $\mathbf{H}_{i1} = \vec{\lambda}_{H_i}^\top \vec{\alpha}_{H_i}$.

Constructing $\mathbf{N} = (\mathbf{N}_0, \mathbf{N}_1)$ requires additional effort as the matrices $\mathbf{N}_0, \mathbf{N}_1$ are no longer diagonal. The decision vector $\vec{\alpha}_N \in \mathbb{R}^{n-1}$ is analogous to $\vec{\alpha}_{H_k}$ and thus subject to the same constraints: $\mathbf{0} \leq \vec{\alpha}_N$ and $\vec{\alpha}_N \mathbf{1} \leq 1$. Finally, $\mathbf{0} \leq \vec{\lambda}_N \in \mathbb{R}^{n^2}$ records both the off-diagonal elements of $\mathbf{N}_0$ and the exit rates $\vec{t}$. Here, $\mathbf{N}_1 = \vec{t} \vec{\alpha}_N$, and the diagonal elements of $\mathbf{N}_0$ are set to ensure the row sums of $\mathbf{Q}_N = \mathbf{N}_0 + \mathbf{N}_1$ sum to 0 as required.

5.4.2 Outline of the fitting algorithm

During the fitting process, the assumption is that we have information on the arrival process $\mathbf{A}$ and a finite number t of marginal queue-length probabilities $\vec{\pi} = (\pi_0, \pi_1, \dots, \pi_t)$. The marginal queue length probabilities are calculated from the queue-length probabilities by summing the

Algorithm 1 Calculating the approximate queue-length probabilities**Input:** $\vec{\alpha}_N, \vec{\lambda}_N, \vec{\alpha}_{H_1}, \vec{\lambda}_{H_1}, \dots, \vec{\alpha}_{H_J}, \vec{\lambda}_{H_J}, \mathbf{A}$

- 1: Construct the PH distributions from $\vec{\alpha}_i, \vec{\lambda}_i$ parameters as shown in Section 5.4.1
- 2: Calculate $\mathbf{F}$ and $\mathbf{L}$ from the components (2.3)
- 3: Solve the first component queue CQ_N with cyclic reduction to get $\mathbf{G}_N$ $\triangleright \mathcal{O}(LCRn^3)$
- 4: Calculate the SVD decomposition of $\mathbf{F}_N \mathbf{G}_N = \mathbf{U}_N \Sigma \mathbf{V}_N$ $\triangleright \mathcal{O}(n^3)$
- 5: Calculate $\mathbf{U}$ and $\mathbf{V}$ for decomposition $\mathbf{F} \tilde{\mathbf{G}} = \mathbf{U} \Sigma \mathbf{V}$ (5.16) $\triangleright \mathcal{O}(m^2)$
- 6: Calculate $\tilde{\pi}_0$ from the components (5.3.4)
- 7: Run Algorithm 2 with input $\mathbf{F}, \mathbf{L}, \tilde{\pi}_0, \mathbf{U}, \Sigma, \mathbf{V}$ to get $\tilde{\pi}_1, \dots, \tilde{\pi}_t$ $\triangleright \mathcal{O}(m^2 t)$
- 8: Calculate marginal probability for level 0 as $\tilde{\pi}_0 = \tilde{\pi}_0 \mathbf{1}_m$
- 9: **return** $\tilde{\pi} = (\tilde{\pi}_0, \dots, \tilde{\pi}_t)$

Algorithm 2 Calculating $\tilde{\pi}_1, \dots, \tilde{\pi}_t$ **Input:** $\mathbf{F}, \mathbf{L}, \tilde{\pi}_0, \mathbf{U}, \Sigma, \mathbf{V}$,

- 1: $\Sigma^{-1} = \text{inv}(\Sigma)$ $\triangleright \mathcal{O}(k)$ as Σ is diagonal
- 2: $\mathbf{L}^{-1} = \text{inv}(\mathbf{L})$ $\triangleright \mathcal{O}(m^2)$ when $\mathbf{L}$ is upper triangular
- 3: $\mathbf{A} = \mathbf{V} \mathbf{L}^{-1}$ $\triangleright \mathcal{O}(km^2)$ as $\mathbf{V}$ is rectangular
- 4: $\mathbf{B} = \Sigma^{-1} + \mathbf{A} \mathbf{U}$ $\triangleright \mathcal{O}(k^2 m)$ as $\mathbf{U}$ is rectangular
- 5: $\tilde{\pi}_0 = (1 - \rho), \quad \tilde{\pi}_{l-1} = \tilde{\pi}_0$
- 6: **for** $l = 1, \dots, t$ **do**
- 7: $\mathbf{u}_0 = \tilde{\pi}_{l-1} (-\mathbf{F}) \mathbf{L}^{-1}$ $\triangleright \mathcal{O}(m^2)$
- 8: $\mathbf{u}_1 = \mathbf{u}_0 \mathbf{U}$ $\triangleright \mathcal{O}(km)$ as $\mathbf{U}$ is rectangular
- 9: $\mathbf{u}_2 = (\mathbf{B}^\top \setminus \mathbf{u}_1^\top)^\top$ $\triangleright \mathcal{O}(k^3)$ to solve system of linear equations
- 10: $\mathbf{u}_2 = \mathbf{u}_2 \mathbf{V}$ $\triangleright \mathcal{O}(km)$ as $\mathbf{V}$ is rectangular
- 11: $\mathbf{u}_1 = \mathbf{u}_2 \mathbf{L}^{-1}$ $\triangleright \mathcal{O}(m^2)$
- 12: $\tilde{\pi}_l = \mathbf{u}_0 - \mathbf{u}_1, \quad \tilde{\pi}_l = \tilde{\pi}_l \mathbf{1}_m, \quad \tilde{\pi}_{l-1} = \tilde{\pi}_l$
- 13: **end for** $\triangleright \mathcal{O}((m^2 + k^3)t)$
- 14: **return** $\tilde{\pi}_1, \dots, \tilde{\pi}_t$

probability vectors at each level, that is $\pi_0 = \tilde{\pi}_0 \mathbf{1}_m$ and $\pi_1 = \tilde{\pi}_1 \mathbf{1}_m$. We start by implementing the equations of Section 5.3 in Algorithm 1, to calculate the approximate marginal queue-length distribution from the decision variables $\vec{\alpha}_N, \vec{\lambda}_N, \vec{\alpha}_{H_1}, \vec{\lambda}_{H_1}, \dots, \vec{\alpha}_{H_J}, \vec{\lambda}_{H_J}$, and the arrival process $\mathbf{A}$. The time complexity of the main steps is noted inline and discussed in detail in Section 5.4.5. Once the relevant matrices and $\tilde{\pi}_0$ are calculated, Algorithm 1 calls Algorithm 2 to calculate the remaining marginal probabilities iteratively. To reduce the time complexity of the computation, we do not compute the matrix $\mathbf{R}$ directly. Instead, the Woodbury identity is applied at every iteration, and matrix-matrix multiplication is entirely avoided. We point to Section 5.4.5 for a detailed analysis of the time complexity corresponding to the steps in Algorithm 2.

The optimisation problem to find a QBD process with a KPC-PH service distribution, that has the given arrival process ($\mathbf{A}$), and that when solved has the given marginal queue length probabilities (π_l for $l = 1, \dots, t$), can be written as follows

$$\begin{aligned} \min \quad & \sum_{l=1}^t |\tilde{\pi}_l - \pi_l| / \pi_l \\ \text{s.t.} \quad & \tilde{\alpha}_i \mathbf{1} \leq \mathbf{1}, \quad i \in I := \{N, H_1, \dots, H_J\} \\ & \tilde{\alpha}_i, \tilde{\lambda}_i \geq \mathbf{0}, \quad i \in I \end{aligned} \tag{5.22}$$

where $\tilde{\alpha}_N \in \mathbb{R}^{n-1}$, $\tilde{\alpha}_{H_i} \in \mathbb{R}^{h_i-1}$, $i = 1, \dots, J$ and $\tilde{\lambda}_N \in \mathbb{R}^{n \times n}$, $\tilde{\lambda}_{H_i} \in \mathbb{R}^{h_i}$, $i = 1, \dots, J$ are the decision variables. Furthermore, $\tilde{\pi} = f(\tilde{\alpha}_N, \tilde{\lambda}_N, \tilde{\alpha}_{H_1}, \tilde{\lambda}_{H_1}, \dots, \tilde{\alpha}_{H_J}, \tilde{\lambda}_{H_J} | \mathbf{A})$ where function f is the implementation of Algorithm 1. The vector $\tilde{\pi}$ and the arrival process $\mathbf{A}$ are inputs to the problem and remain unchanged throughout. In the case of an irreducible Markov chain, $\tilde{\pi} > 0$; hence, division by π_l is always allowed up to machine precision. Therefore, we have $V = n - 1 + n^2 + \sum_{i=1}^J (2h_i - 1)$ variables and $C = V + n - 1 + \sum_{i=1}^J (h_i - 1)$ constraints. The objective function calculates the error between the input $\tilde{\pi}$ and approximate $\tilde{\pi} = (\tilde{\pi}_0, \dots, \tilde{\pi}_t)$. Normalizing the difference forces the fitting algorithm to correctly capture the tail of the distribution. This is necessary because the probabilities corresponding to higher levels are very close to zero; therefore, otherwise, the difference between the approximate and the input probabilities would be negligible.

The optimisation problem (5.22) follows a common form with linear constraints and a non-linear objective function. A broad range of solvers and metaheuristics are available that may be used in principle to search for an optimal solution. Details of our implementation are presented in Section 5.4.6. Due to the linear constraints, the challenging aspect of the optimisation problem is the objective function. In Section 5.4.5, we demonstrate that our theoretical results reduce the complexity of the objective function from cubic to quadratic. This computational gain can be leveraged by any chosen solver. Due to the potential noise in the trace, the possibility that the fitted distribution might not exactly follow a PH distribution, and the use of approximate queue-length probabilities in the objective function, an exact solution to the optimisation problem is not feasible. Instead, the goal is to find a PH distribution that is a local optimum and provides a good approximation to the input queue-length probabilities.

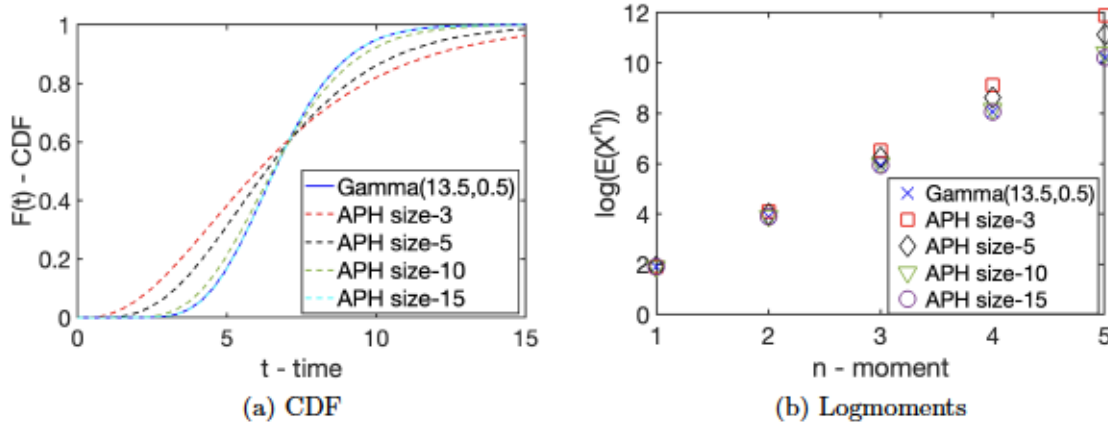


Figure 5.1: Gamma distribution with $SCV = 0.074$ approximated by APH distributions of increasing size.

5.4.3 Using a gamma distribution as N

As noted, the general PH distribution N requires more parameters than the hyperexponential components H_k . To reduce the number of parameters and therefore the dimension of the optimisation problem, we note the following. The SCV of N can be chosen to be $SCV_N \leq 1$ without loss of generality. This is because a PH distribution with $SCV > 1$ is hyperexponential. Setting $SCV_N > 1$ would mean the KPC-PH distribution is hyperexponential [Neu94]. However, setting $SCV_N = 1$ gives a hyperexponential KPC-PH as well. In this case, the first component is lost, but the number of hyperexponential components can be increased from J to $J + 1$ to counteract the lost component.

During the optimisation, we reduce the dimension of the search problem by using a gamma distribution, which requires only two parameters. At each iteration, the gamma distribution is approximated with an acyclic phase-type (APH) distribution. This approach is natural since the gamma distribution generalizes the Erlang distribution by allowing a non-integer shape parameter.

Figure 5.1 shows that a gamma distribution with an SCV as low as .074 is closely matched by an APH distribution of size 10, and is matched perfectly by an APH distribution of size 15.

Algorithm 3 Reducing a large KPC-QBD distribution.**Input:** S_0, S_1, k

- 1: $Q := S_0 + S_1$ ▷ Calculate infinitesimal generator
- 2: Solve $\vec{\pi}Q = 0, \vec{\pi}\mathbf{1} = 1$ for $\vec{\pi}$ ▷ Find steady-state distribution of CTMC
- 3: $A_k := \text{maxIndex}(\vec{\pi}, k)$ ▷ Find indices of k largest probabilities in $\vec{\pi}$
- 4: $S_0^{(k)} := S_0|_{A_k}$ ▷ $S_0^{(k)}$ is the submatrix corresponding to states with k -largest probabilities
- 5: Solve $S_1 = -S_0\mathbf{1}\vec{\alpha}$ for $\vec{\alpha}$ ▷ Calculate original starting probabilities
- 6: $\vec{\alpha}^{(k)} := \frac{\alpha|_{A_k}}{\|\alpha|_{A_k}\|_1}$ ▷ Restrict and normalize α
- 7: $S_1^{(k)} := -S_0^{(k)}\mathbf{1}\vec{\alpha}^{(k)}$ ▷ Calculate $S_1^{(k)}$ from $S_0^{(k)}$ and $\vec{\alpha}^{(k)}$
- 8: **return** $S_0^{(k)}, S_1^{(k)}$

The updated version of (5.22) that uses a two-parameter gamma distribution is as follows

$$\begin{aligned}
\min \quad & \sum_{l=1}^t |\tilde{\pi}_l - \pi_l| / \pi_l \\
\text{s.t.} \quad & \vec{\alpha}_i \mathbf{1} \leq 1, \quad i \in I := \{H_1, \dots, H_J\} \\
& k, \theta, \vec{\alpha}_i, \vec{\lambda}_i \geq \mathbf{0}, \quad i \in I
\end{aligned} \tag{5.23}$$

where $\vec{\pi} = f(k, \theta, \vec{\alpha}_{H_1}, \vec{\lambda}_{H_1}, \dots, \vec{\alpha}_{H_J}, \vec{\lambda}_{H_J} | \mathbf{A})$ and function f is updated to include the conversion of the gamma distribution into an APH distribution before the queue-length probabilities are computed. The parameters (k, θ) correspond to the gamma distribution.

5.4.4 Reducing the size of the fitted distribution

When using a gamma distribution during the fitting algorithm, the size of the final KPC-PH distribution can become large. This occurs because, at each iteration, the gamma distribution is approximated with an APH distribution. However, in the case of a gamma distribution with $SCV < 1$, this last step requires at least $1/SCV$ number of states in the first component, since a PH distribution with n states cannot achieve an SCV lower than $1/n$ [AS87].

Managing a PH distribution with several thousand states is computationally challenging in analytical models and a large number of phases can also indicate overparameterization. To address this problem, we introduce a reduction algorithm outlined in Algorithm 3. The algorithm always produces a valid PH distribution by construction. The method works by keeping the phases corresponding to the k -largest steady-state probabilities. Parameter k can be chosen to ensure the reduced distribution closely matches the accuracy of the original distribution. While

overparameterization might not be a significant issue — since noise from the trace primarily affects the tail of the probability distribution, which is hard to fit accurately — a reduced PH distribution size ensures a smoother fit. This smoothing effect occurs because the reduced distribution has fewer degrees of freedom, mitigating any overfitting that may have occurred due to overparameterization.

5.4.5 Time and space complexity analysis

The running times of the most time-consuming operations are noted inline in Algorithms 1 and 2. We give a summary of the time complexity analysis here. Using the same notation as before, let n be the size of the first PH component of the KPC-PH, k be the rank of matrix $\mathbf{F}_N \mathbf{G}_N$, and m the size of the entire KPC-PH distribution. To give an upper bound on the running times of Algorithms 1 and 2, we work under the assumption that $n \ll m$. It can be seen that Algorithm 1 runs in $\mathcal{O}(m^2)$ time plus the time it takes to run Algorithm 2. To provide a theoretical upper bound on Algorithm 2, we further assume that matrix $\mathbf{L}$ is upper triangular. This assumption holds if both the arrival PH distribution and the first component of the KPC distribution are Coxian. However, in practice, we found that the inversion of matrix $\mathbf{L}$ in line 2 is much faster than $\mathcal{O}(m^2)$ due to its sparsity. Algorithm 2 is dominated by the for loop that runs in $\mathcal{O}(m^2 + k^3)t$ time. Since $k < n \ll m$, $\mathcal{O}((m^2 + k^3)t) = \mathcal{O}(m^2t)$. Therefore, Algorithm 1 runs in $\mathcal{O}(m^2t)$ time. Using the cyclic reduction method to solve the original QBD model directly runs in $\mathcal{O}(L_{CR}m^3)$ time where L_{CR} is the number of iterations the algorithm requires to converge, with $\mathcal{O}(m^3)$ being the dominating factor at every iteration. Calculating $\mathbf{R}$ from $\mathbf{G}$ and generating t marginal probabilities takes $\mathcal{O}(m^3 + tm^2)$ time. Thus, the naive method runs in $\mathcal{O}(L_{CR}m^3 + tm^2)$ time. Therefore, our proposed method reduces the time complexity from $\mathcal{O}(L_{CR}m^3 + tm^2)$ to $\mathcal{O}(m^2t)$.

In terms of space complexity, all steps in both Algorithm 1 and 2 are dominated by the space required to store the inputs of the corresponding operations. The operations for which this is not a straightforward claim are: the cyclic reduction algorithm in Algorithm 1 which requires $\mathcal{O}(n^2)$ space; the SVD decomposition in Algorithm 1, which requires $\mathcal{O}(n^2)$ space; and solving

a system of linear equations in Algorithm 2, which needs $\mathcal{O}(k^2)$ space. Therefore, an overall upper bound for the space complexity is $\mathcal{O}(m^2)$, the dimension of the largest matrices involved, such as L and F . However, because these matrices exhibit a Kronecker product and Kronecker sum structure, they are inherently sparse and can be stored as sparse matrices. As a result, the actual space complexity tends to be lower than the $\mathcal{O}(m^2)$ upper bound.

5.4.6 Details of the solver used for the optimisation problem

In our implementation, we used MATLAB and its built-in solver `fmincon` [Mat24] to solve the optimisation problem 5.23. The `fmincon` solver offers multiple algorithms for solving a non-linear optimisation problem with linear constraints. When it comes to fitting a QBD process, an exact solution is impractical. Firstly, the observed trace or simulation data the method aims to fit is not exact. Secondly, even with precise data on the system, the PH distribution used for the fitting only approximates the real distribution of the system — arbitrarily well in theory, but with some error in practice. Lastly, our method introduces an additional layer of approximation, as our proposed algorithm calculates approximate marginal queue-length probabilities. Therefore, we choose the interior-point method [BHN99, BGN00] of the `fmincon` solver, as it promotes speed over accuracy, which is well-suited to our problem where an exact solution is unattainable. Furthermore, the interior-point algorithm is a large-scale method, meaning it does not need to store or operate on full matrices internally. As discussed earlier, the large matrices in our algorithm are sparse; therefore, supporting sparse matrix operations ensures that memory usage is kept as low as possible. The numerical tests were run on an Ubuntu machine with an Intel Xeon E5540 CPU having 16 virtual cores and 32GB of RAM.

5.5 Numerical results

In this section, we conduct numerical tests to evaluate the proposed divide-and-conquer algorithm, both in isolation and as part of a search algorithm for fitting. In Section 5.5.1, we compare the speed and accuracy of Algorithm 1 with a benchmark cyclic reduction method for calculating marginal queue-length probabilities for a QBD process with specified arrival and

Table 5.2: Comparing a cyclic reduction method to the proposed algorithm for inputs with large SCV values. (ARV=Erlang-5 with mean of 2; $S = N \otimes H_1 \otimes \dots \otimes H_J$ where $N = \text{APH}(1, 10)$, $H_i = \text{HE}(1, 1.1)$)

size m	J	SCV	Timings(s)		Error	
			Exact	Algorithm 1	avg norm-1	rel norm-2
40	3	11.127	0.0218 ± 0.0224	0.0217 ± 0.0184	0.0000	0.0015
80	4	11.734	0.0126 ± 0.0018	0.0116 ± 0.0002	0.0000	0.0022
160	5	12.371	0.1085 ± 0.0199	0.0296 ± 0.0011	0.0000	0.0029
320	6	13.039	0.2879 ± 0.0515	0.0673 ± 0.0011	0.0000	0.0035
640	7	13.741	1.0983 ± 0.1796	0.1135 ± 0.0239	0.0000	0.0042
1280	8	14.478	5.3093 ± 0.1354	0.1774 ± 0.0093	0.0000	0.0048
2560	9	15.252	47.392 ± 0.8600	0.3872 ± 0.0120	0.0000	0.0055
5120	10	16.065	326.63 ± 21.585	0.7134 ± 0.0199	0.0000	0.0061
10240	11	16.918	2148.1 ± 89.825	2.8448 ± 0.0657	0.0000	0.0067

service distributions. In Section 5.5.2, we test the fitting algorithm that utilises the speed-up of Algorithm 1, first against a fitting algorithm that uses the benchmark cyclic reduction method in its objective function, and then against other fitting algorithms that fit traces directly. In Section 5.5.3, we show that the improved fit translates to increased response time predictability. Finally, the advantages and limitations of our method are discussed in Section 5.5.4.

5.5.1 Evaluating Algorithm 1

To assess the computational efficiency of Algorithm 1, we run large input tests using a benchmark cyclic reduction algorithm implemented in Q-MAM [PVVH08]. Several improvements are made to the cyclic reduction method to address implementation-related bottlenecks (e.g., preallocating memory to prevent arrays from changing size) and to make the comparison more meaningful.

Tables 5.2 and 5.3 show the results for inputs with large and small SCV values, respectively. In both tables, m refers to the number of phases in the process, that is the size of the matrices used in the calculations, and J is the number of hyperexponential components that make up the KPC service process. In both cases, we used an Erlang-5 distribution with a mean of 2 as the arrival process. The mean of the service processes is kept at 1 resulting in a 50% utilisation. For the high SCV tests, the first component of the KPC is an APH with mean 1 and SCV 10, while the hyperexponential components have mean 1 and SCV 1.1. For the low SCV tests,

Table 5.3: Comparing a cyclic reduction method to the proposed algorithm for inputs with small SCV values. (ARV=Erlang-5 with mean of 2; $S = N \otimes H_1 \otimes \dots \otimes H_J$, $N=APH(1, 0.1)$, $H_i = HE(1, 1.1)$)

Size m	J	SCV	Timings (s)		Error	
			Exact	Algorithm 1	avg norm-1	rel norm-2
40	3	0.2127	0.0440 ± 0.0146	0.0059 ± 0.0023	0.0004	0.0096
80	4	0.2734	0.1559 ± 0.0130	0.0057 ± 0.0008	0.0002	0.0141
160	5	0.3371	0.9574 ± 0.1265	0.0108 ± 0.0018	0.0001	0.0185
320	6	0.4039	5.9840 ± 0.1849	0.0241 ± 0.0032	0.0001	0.0228
640	7	0.4741	51.864 ± 2.7890	0.0906 ± 0.0130	0.0000	0.0269
1280	8	0.5478	485.82 ± 27.930	0.2297 ± 0.0170	0.0000	0.0309
2560	9	0.6252	3368.8 ± 351.37	0.7689 ± 0.0376	0.0000	0.0348

the first component is updated to have SCV 0.1, with all remaining inputs unchanged. The algorithms have been run 10 times, and the average running times are presented with their corresponding 95% confidence intervals. During the KPC algorithm, marginal probabilities are calculated for the same number of levels for which the cyclic reduction produced values. The difference between the two vectors is calculated and displayed in the last column of the tables. The reduction in running times is significant, even for relatively smaller sizes. In alignment with theoretical expectations discussed in Section 5.4.5 that the running time decreases from cubic to quadratic, these experiments demonstrate that the divide-and-conquer method significantly accelerates the calculation of marginal queue-length probabilities while maintaining a small error in the approximation.

5.5.2 Evaluating the fitting algorithm

In this section, we use two real-life traces, BuildServer00¹ and Radius Authentication² to evaluate the fitting algorithm outlined in (5.22) and implemented in MATLAB. In the upcoming figures, we refer to our method as *KPC-QBD*. We first compare *KPC-QBD* to an alternative queue-based fitting method, which we term *exact QBD*. The alternative approach involves using optimisation problem (5.22); however, instead of the divide-and-conquer algorithm we developed, it employs the cyclic-reduction algorithm implemented in QMAM to compute the queue-length probabilities. We also employ other well-established fitting algorithms as bench-

¹<http://iota.snia.org/traces/block-io/158?n=10&page=1>

²<http://iota.snia.org/traces/block-io/158?n=10&page=2>

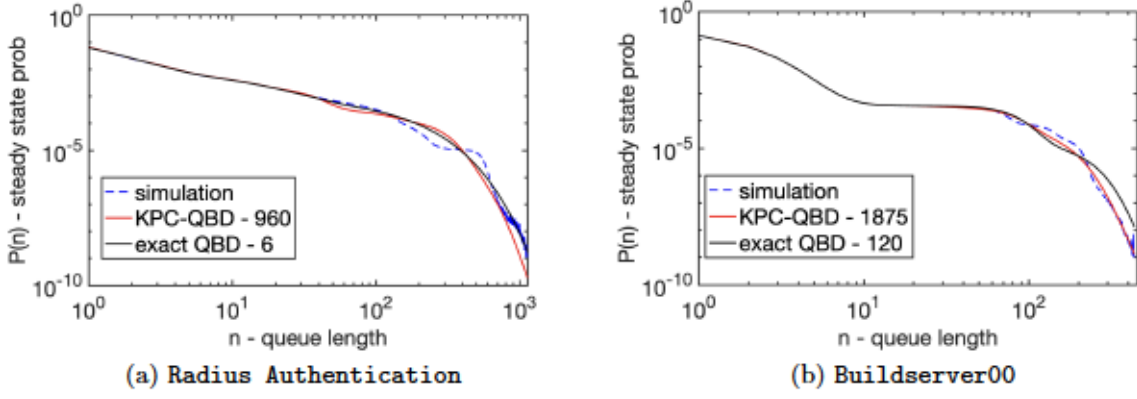


Figure 5.2: Queueing results at 25% utilisation when using KPC-QBD and exact QBD methods to fit queueing probabilities.

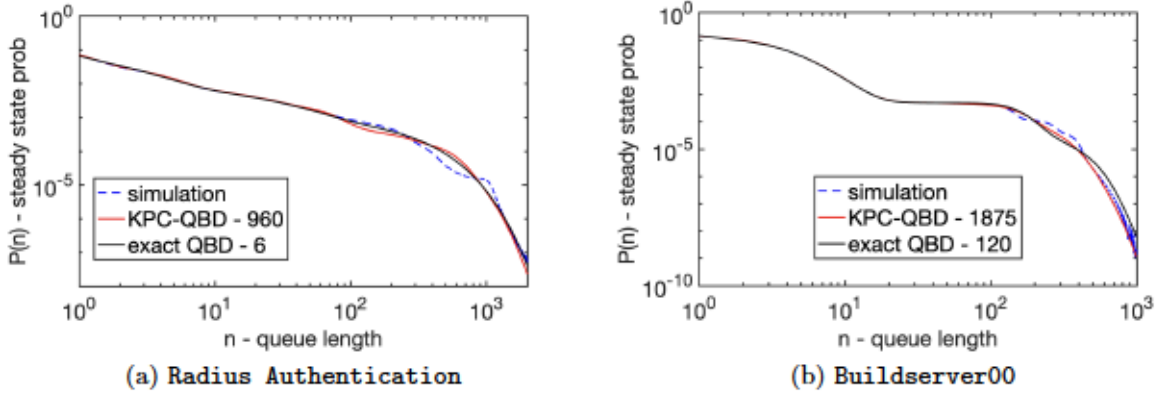


Figure 5.3: Queueing results at 50% utilisation when using KPC-QBD and exact QBD methods to fit queueing probabilities.

marks, all of which directly fit traces using moments of the trace. These fitting algorithms are implemented in the KPC toolbox [CZS10a] and include:

- *Prony's method*, which utilises characteristic polynomials to exactly match moments of the trace (see, [CZS10b, Theorem 1], and [CS12]).
- *Hyperexponential parameter fitting*, an approximate moment-matching algorithm that uses a non-linear optimisation problem to find the hyperexponential distribution that best approximates the first three moments of the empirical trace.
- *KPC-PH algorithm* [CZS10a] that uses a divide-and-conquer approach to match moments of a trace directly.

Initially, when examining only marginal queue-length probabilities in Figure 5.2, the difference between *KPC-QBD* and *exact QBD* appears small. Fitting is performed at the 25% utilisation

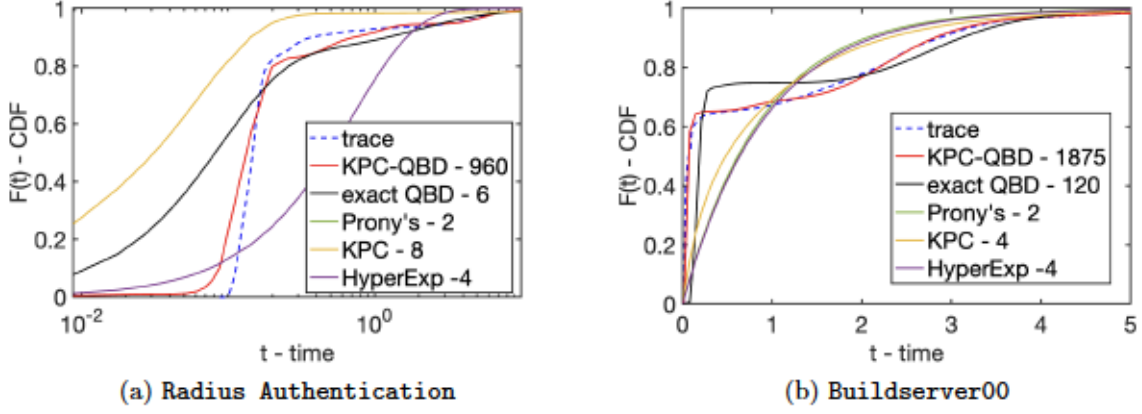


Figure 5.4: Empirical CDF of traces and the corresponding fitted distributions.

level where some improvement is noticeable. At higher utilisation levels (see Figure 5.3), the sensitivity to the accuracy of the service distribution fitting diminishes. However, a significant difference in fitting accuracy becomes evident upon inspecting the empirical Cumulative Distribution Function (CDF) of the traces directly in Figure 5.4. The Kolmogorov-Smirnov distance between the fitted distributions and the trace is also reduced; details are given in Table 2 in the appendix. *KPC-QBD* not only outperforms the exact QBD method but also surpasses all other benchmark methods that directly fit the traces. Noticeably, the sizes of the PH distributions fitted by the *exact QBD* method are significantly smaller compared to those produced by *KPC-QBD*. This size difference is an anticipated limitation of *exact QBD*, attributable to the slow computation of cyclic reduction at each iteration of the optimisation problem while evaluating its objective function.

5.5.3 Response time distributions

We now demonstrate how the improved fit of the service distribution translates to more accurate response time distribution estimations, further validating the proposed *KPC-QBD* algorithm. We calculate the response time CDFs at 25% and 50% utilisation for both *Buildserver00* and *Radius Authentication* traces. There are toolboxes that provide solvers for response time calculations such as [PVVVH08]. In Figures 5.5 and 5.6, the response time CDFs are computed, except for *KPC-QBD* distributions, which are simulated due to their large number of states. It is apparent from these figures that the *KPC-QBD* distributions offer a much closer

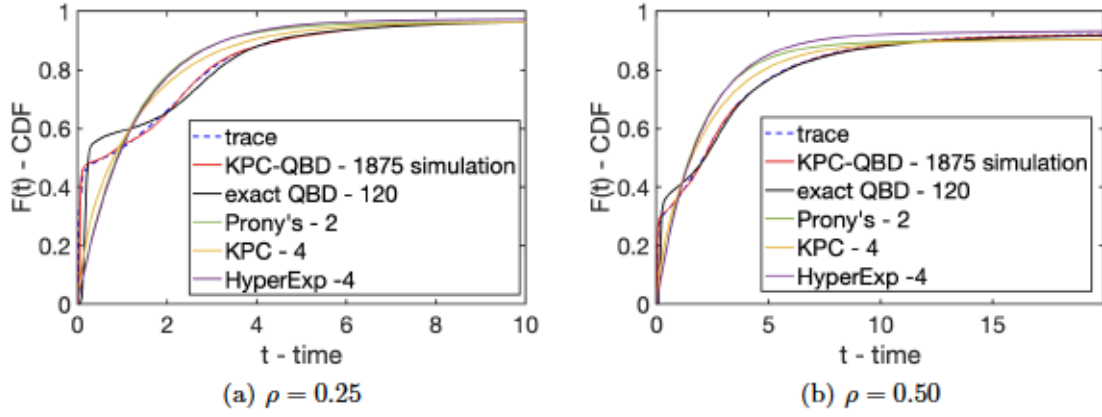


Figure 5.5: Buildserver00 trace CDF of response time, KPC-QBD and benchmark methods.

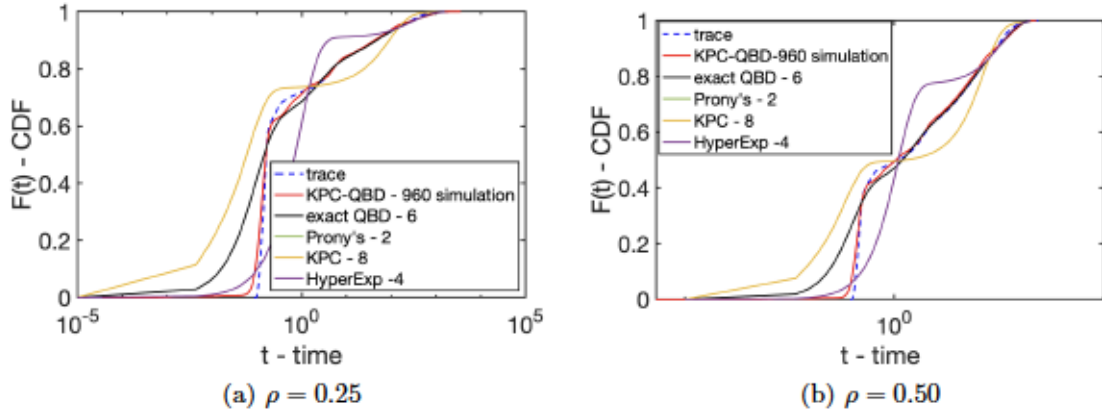


Figure 5.6: Radius Authentication trace CDF of response time, KPC-QBD and benchmark methods.

fit compared to the other methods, including the *exact QBD* distributions. The Kolmogorov-Smirnov distance between the response time CDF of the fitted distributions and the trace is significantly reduced as reported in the appendix in Table 3 at 25% utilisation and Table 4 at 50% utilisation.

Motivated by the accuracy of the *KPC-QBD* method, our next goal is to reduce the size of the *KPC-QBD* distributions to make response time calculations numerically feasible, while making sure the accuracy of their fit remains high. The reduction is achieved by running the algorithm introduced in Section 5.4.4 (see Algorithm 3). First, on the left of Figure 5.7, we run simulations of varying lengths to calculate the complementary CDF (CCDF) of the response time distribution for the Buildserver00 trace. The results stabilise at 10^6 iterations, producing the same CCDF as 10^7 iterations. Running the simulation for 10^6 iterations takes

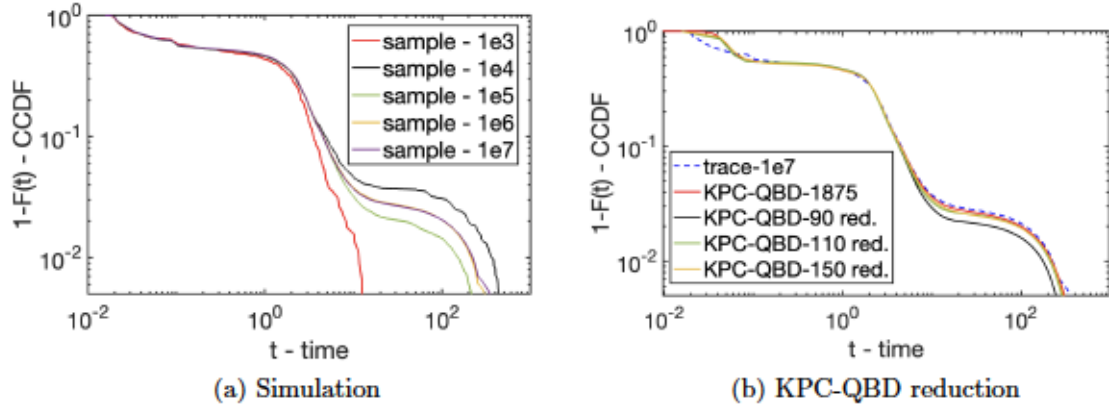


Figure 5.7: Buildserver00 response time CCDF at 25% utilisation, comparing simulation to reduction.

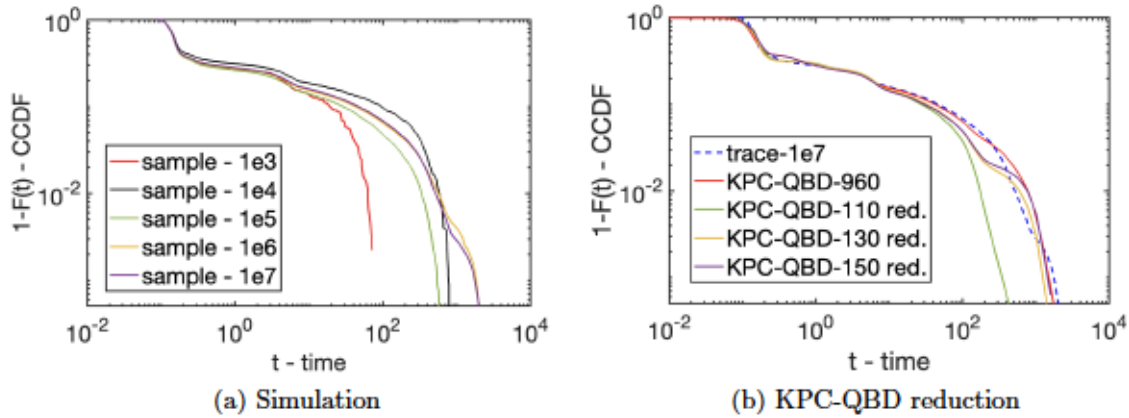


Figure 5.8: Radius Authentication response time CCDF at 25% utilisation, comparing simulation to reduction.

approximately 38 seconds. The simulation of length 10^6 is then compared with the original and reduced *KPC-QBD* distributions on the right. A reduced *KPC-QBD* of size 110 maintains good accuracy and it takes 10 seconds to compute the PH distribution corresponding to the response time, achieving a 4-fold speed-up compared to simulation. The CCDF is shown as it highlights the tail of the response time distribution, which is the most sensitive to inaccuracies introduced by the reduction algorithm. Similarly, in the case of the Radius Authentication trace depicted in Figure 5.8, the simulation requires approximately 10^6 iterations for the CDF to stabilise. The reduced *KPC-QBD* on the right requires a size of 130 for a reasonable estimate, running in approximately 27 seconds.

5.5.4 Advantages and limitations of the proposed algorithm

Both queue-based fitting methods outperform the benchmark methods. This is likely because those methods only use moments for the fitting; however, a few moments do not fully capture the underlying distributions. Furthermore, as a result of the speed up in calculating marginal queue-length probabilities with the proposed algorithm, we can perform queue-based fitting without the significant limitations on the size of the fitted distribution the cyclic reduction method imposes. This increases the accuracy of the fitted distribution. Once a potentially large closely fitting distribution is found, we are able to reduce the size of the distribution whilst maintaining the accuracy of its fit, producing a final distribution that is feasible to use in analytical tools performing predictive analyses, such as obtaining response time measures.

The limitations of this method include the fact that we require the QBD process fitted to be in a steady state, which does not apply to all QBD processes. Additionally, while the divide-and-conquer approach addresses some size limitations of the fitted distributions, certain distributions may still necessitate PH distributions with even more states. Lastly, PH distributions are unable to capture correlations in service time distributions.

5.6 A compositional ME fitting algorithm

Motivated by the accuracy of the fitted distributions in the previous section, we explore fitting with ME distributions introduced in Section 2.3. The QBD results from Section 5.3 no longer apply. However, by extending the compositional approach used to construct the KPC-PH distribution, we develop a novel ME fitting algorithm that preserves the compositional properties of KPC-PH distributions. Specifically, it enables efficient calculation of the CDF and Mellin transform through decomposition. Our proposed method can generate high-order ME distributions while ensuring a valid density function throughout the fitting process. Theoretical results on how the proposed family of ME mixtures is constructed are presented in Section 5.6.1. Section 5.6.2 defines the fitting algorithm. Numerical results are presented in Section 5.6.3.

5.6.1 Mixture of ME distributions

The family of distributions used for the fitting is constructed as the product of a valid ME distribution of low order and a finite number of independent random discrete scaling factors. Even though the original ME distribution has few degrees of freedom, the scaling factors ensure that the resulting ME mixture is significantly more flexible. Due to this increased flexibility and the fact that it is a product of independent random variables, each of which is easy to control in isolation, the constructed ME mixture is an excellent candidate for fitting. We start by formally defining the random scaling factors.

Definition 5.3. *Let C be a random variable that takes values $\vec{\lambda} = (\lambda_1, \dots, \lambda_k) \in \mathbb{R}$ with respective probabilities $\vec{p} = (p_1, \dots, p_k)$, where $\sum_{i=1}^k p_i = 1$ and $p_i \in [0, 1]$. Then $C \sim \text{CAT}(\vec{\lambda}, \vec{p})$ is called a categorical distribution, .*

Proposition 5.6.1. *Let $X = YC_1 \dots C_m$, where Y and C_i are independent random variables, $i = 1, \dots, m$. Furthermore, $Y \sim \text{ME}(\vec{\alpha}, \mathbf{A})$ and $C_i \sim \text{CAT}(\vec{\lambda}^{(i)}, \vec{p}^{(i)})$, where $\vec{\lambda}^{(i)} > 0$ $i = 1, \dots, m$. Then X is a finite mixture of matrix exponential distributions.*

Proof. The product of independent categorical distributions is categorical; therefore, it is enough to prove for $X = YC$ where $C \sim \text{CAT}(\vec{\lambda}, \vec{p})$ and $\vec{\lambda}$ is a strictly positive vector. Conditioning on the values of C , the conditional distribution is $X|\{C = \lambda_j\} = Y\lambda_j \sim \text{ME}(\vec{\alpha}, \frac{1}{\lambda_j}\mathbf{A})$. Therefore, X is the mixture of ME distributions, where each ME in the mixture is a scaled version of Y . $\square$

An immediate consequence of Proposition 5.6.1 is that X is matrix exponentially distributed since the class of ME distributions is closed under finite mixtures [BN17]. The proposition above assumes that Y has a valid ME distribution. As stated before, there is no efficient way to check if a vector $\vec{\alpha}$ and a matrix $\mathbf{A}$ produce a valid non-negative density function. In order to overcome this, we restrict Y to a subclass of matrix exponential distributions which produces a valid density by definition [BHT11, HSTZ16]. Here we only include the definition for odd n .

Definition 5.4. *$Y \sim \text{ME}(\omega, \phi)$ with $\omega \in \mathbb{R}$ and $\phi \in R^n$ has a matrix exponential distribution of order $n = 2k - 1$ with unnormalized density $f(t) = e^{-t} \prod_{i=1}^{(n-1)/2} \cos^2(\omega t - \phi_i)$.*

Proposition 5.6.2. *In the special case where Y has a PH distribution, $Y \sim PH(\vec{\alpha}, \mathbf{A})$, X defined as in Proposition 5.6.1 has a KPC-PH distribution,*

$$X \sim Y \otimes C_1^* \otimes \dots \otimes C_m^*$$

where $C_i^* \sim HyperExp(\vec{p}^{(i)}, \mathbf{\Lambda}^{(i)})$ and $\mathbf{\Lambda}^{(i)}$ is a diagonal matrix with $\text{diag}(\mathbf{\Lambda}^{(i)}) = -1/\vec{\lambda}^{(i)}$.

Proof. First, note that X has a phase-type distribution since the class of PH distributions is also closed under finite mixtures [MO92]. To prove that X has a KPC-PH distribution, it is enough to prove that a KPC-PH distribution with parameters given above has Mellin transform $M_Y(s)M_{C_1}(s) \dots M_{C_m}(s)$, that is,

$$M_{KPC-PH}(s) = \Gamma(s)\vec{\alpha}(-\mathbf{A})^{(1-s)}\mathbf{1} \prod_{i=1}^m \vec{p}^{(i)}(-\mathbf{\Lambda}^{(i)})^{(1-s)}\mathbf{1}$$

This can be proven analogously to the proof given for the product of two independent PH distributions in [BNN16][Theorem 4.4]. $\square$

Therefore, the proposed fitting method may be seen as a generalization of the KPC-PH method introduced in [CZS10a].

5.6.2 Fitting algorithm

The next step is to choose the measures based on which the fitting is performed. Defining X as in Proposition 5.6.1 has several advantages. First, the CDF of the chosen subclass of ME distributions have a closed form. Furthermore, the scale factors can only take a finite number of different values. Therefore F_X , the CDF of X , can be calculated efficiently even for a large number of states. Next, since X is a product of independent random variables, $E(X)$ also factorises, or more generally, the Mellin transform, $M_X(s) = E(X^{s-1})$ and $s \in \mathbb{C}$, factorises. This is important as the Mellin transform of an ME distribution has a matrix exponential component. We use the Mellin transform as it allows the calculation of half-moments (i.e. $E(X^{1.5})$) on top of ordinary moments; therefore, higher moments can be avoided by instead

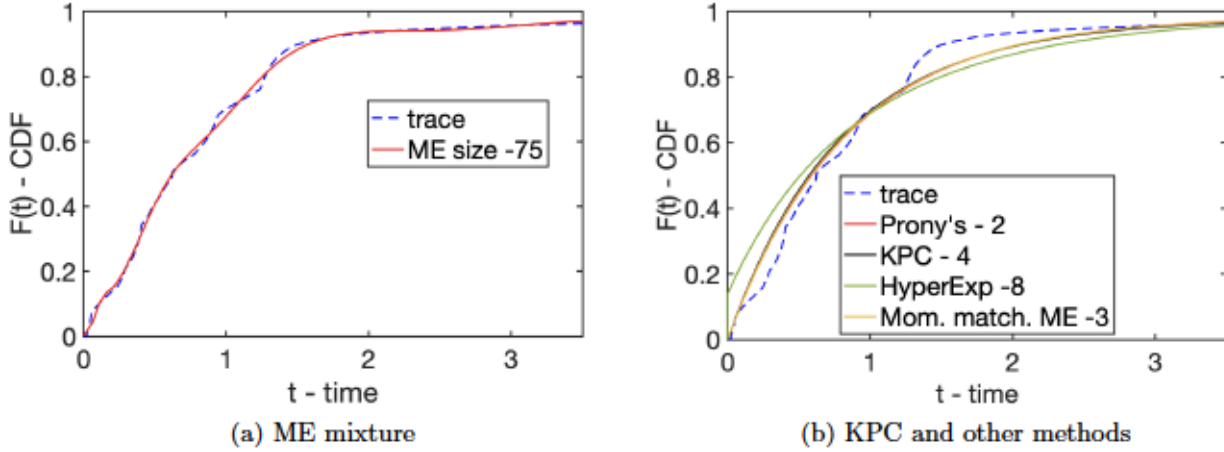


Figure 5.9: ECDF of the BCAug89 trace and CDF of the fitted distributions.

using the first $\lfloor n/2 \rfloor$ moments and half moments for the fitting. Both the empirical CDF (F_{tr}) and Mellin transform (M_{tr}) are easily obtainable from a trace. Therefore the objective function to be minimised during the fitting is

$$C_X(\vec{t}, \vec{s}) = \|F_{tr}(\vec{t}) - F_X(\vec{t})\| + \|M_{tr}(\vec{s}) - M_X(\vec{s})\|$$

where $\vec{t}, \vec{s}$ are the inputs at which the CDF and Mellin transform are calculated, respectively. The optimisation problem is therefore formulated as

$$\begin{aligned} \min \quad & C_X(\vec{t}, \vec{s}) \\ \text{s.t.} \quad & \vec{p}^{(i)} \mathbf{1} \leq \mathbf{1}, \quad 1 \leq i \leq m \\ & \vec{\lambda}^{(i)} \geq \epsilon, \quad 1 \leq i \leq m \end{aligned}$$

where $\epsilon = (\epsilon, \dots, \epsilon)$ with $\epsilon > 0$ to ensure the positivity of the scaling factors, and X is as defined in 5.6.1 with Y taken to be of the form of Definition 5.4. Analogous to Section 5.4.1, $\vec{p}^{(i)}$ has dimension one less than $\vec{\lambda}^{(i)}$ and the missing probability is calculated as $1 - \vec{p}^{(i)} \mathbf{1}$.

5.6.3 Numerical tests

Since the optimization problem is non-convex, the fitting process is run multiple times, and the best distribution is selected for each fitting method. Two traces have been chosen to

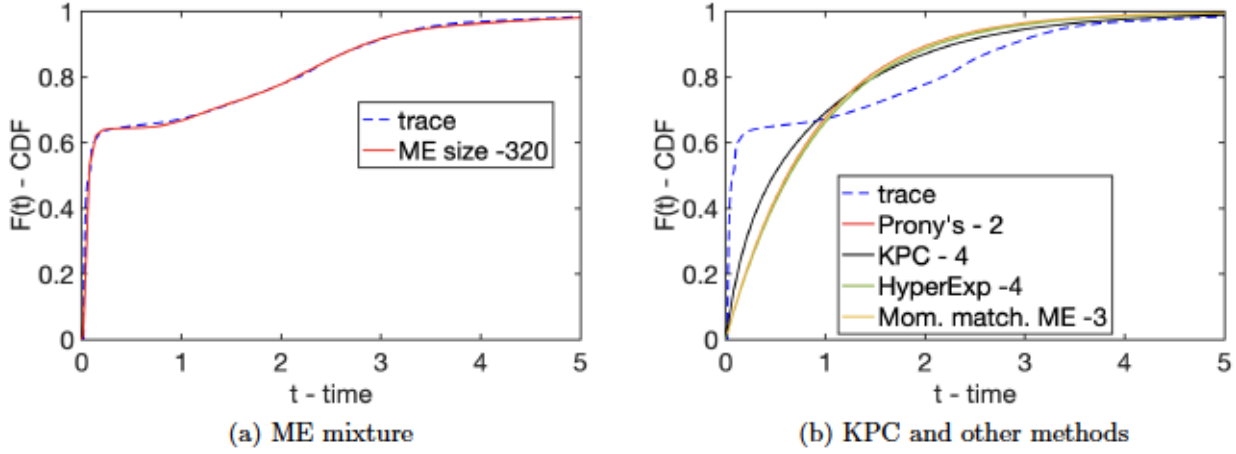


Figure 5.10: ECDF of the BuildServer00 trace and CDF of the fitted distributions.

Table 5.4: Log-moments of BCAug89 trace and ME distributions (fitted with and without the Mellin transform).

$\log(E(X^n))$	Trace	ME with Mellin	ME without Mellin
$n = 1$	0	0	0
$n = 2$	1.44	1.44	24.53
$n = 3$	4.17	4.17	49.58
$n = 4$	7.53	7.56	75.04
$n = 5$	11.32	11.65	100.81

demonstrate the adequacy of the proposed method. The first one is the Bellcore pAug89 (BCAug89) trace which is often used in literature for the evaluation of the accuracy of fitting methods [AN98, HT02], the second one is the BuildServer00 trace³ recorded on 11-28-2007 at 7:55PM. Both traces were randomly shuffled in order to remove temporal dependence.

Figure 5.9 shows the empirical CDF of the BCAug89 trace and the fitted ME distribution on the left 5.9(a), where size-75 refers to the number of states in the distribution. It is compared against the KPC-PH method, moments-based ME fitting [HT16], Prony's method and hyper-exponential parameter fitting on the right 5.9(b). The mixture of ME distributions produces a much closer fit than any of the other methods considered. Figure 5.10 shows the same for the BuildServer00 trace. Once more, the mixture of ME distributions considerably outperforms the other methods.

Next, the ME mixture is fitted based on the empirical Mellin transform and the empirical CDF (ECDF) is compared to the ME mixture that is fitted based on the ECDF alone. Table 5.4

³<http://iota.snia.org/traces/block-io/158>

shows how including the Mellin transform in the fitting makes sure the moments are matched correctly. This in turn improves how well the tail of the distribution is fitted (Figure 5.11). Finally, the fitted ME distributions are sampled and a queueing process is simulated with a fixed hyperexponential arrival process and the fitted distribution as the service process at 25% and 50% utilisations. The simulation first processes the shuffled traces, followed by the corresponding ME mixture samples. Queueing results solved analytically for a fitted KPC-PH and hyperexponential distribution are also displayed for comparison. Figure 5.12 and Figure 5.13 show the steady-state queueing probabilities for the BCAug89 and BuildServer00 traces, respectively. The ME mixture provides a slightly better fit for the tail of the distribution in the case of the BCAug89 trace, while the improvement is significant in the case of the BuildServer00 trace. Figure 5.14 and Figure 5.15 display the response time CDFs for the BCAug89 and BuildServer00 traces, respectively, demonstrating that the ME mixtures accurately capture the response time distributions as well. As expected, the increased accuracy of the fitted distributions translates to the increased accuracy of the queueing probabilities and the response time distributions.

The results highlight the potential of the proposed fitting method. The ME mixture follows the CDF much more closely than any other fitted distribution. This outcome is expected, as our algorithm directly utilises the empirical CDF for fitting rather than relying solely on moments. Additionally, the use of the Mellin transform enables the exclusion of higher moments from the fitting process. The strong improvement in the accuracy of queueing prediction demonstrates the practical advantage of the improved fitting.

Finally, it is interesting to compare the ME fitting algorithm to the KPC-QBD algorithm. Comparing Figure 5.10a to Figure 5.4b, both the KPC-QBD and ME fitting methods achieve highly accurate parameterizations for the BuildServer00 trace. This naturally raises the question of which approach is preferable. While both methods support fitting distributions with many phases due to their compositional structure, they serve different purposes. KPC-QBD can be used even when direct measurements of the service distribution are unavailable, whereas the ME fitting algorithm requires direct measurements to fit the service distribution. On the other hand, KPC-QBD results are applicable only to queues with PH arrivals under the

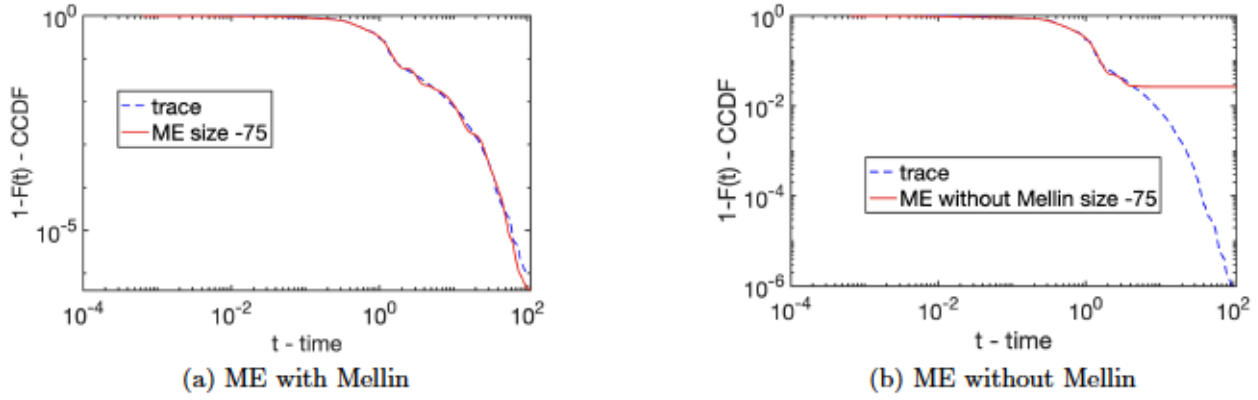


Figure 5.11: Comparing the CCDF of the BCAug89 trace to the ECDF when fitting with and without the Mellin transform.

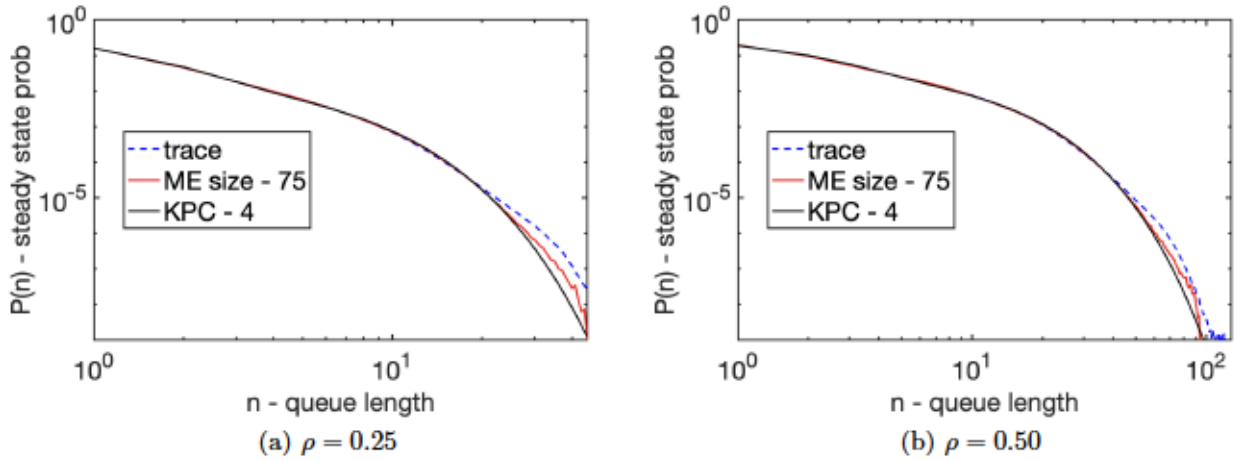


Figure 5.12: Steady state queueing probabilities for BCAug89 trace at 25% and 50% utilisation levels.

FCFS discipline, whereas the ME parameterizations obtained through the ME fitting algorithm can be used in any queueing system. Therefore, the choice of fitting algorithm depends on the specific context.

5.7 Summary

In this chapter, we develop a novel QBD-based fitting algorithm for fitting PH service distributions and design a novel ME fitting method that guarantees the validity of the fitted distribution. The main contributions of the chapter are summarised as follows:

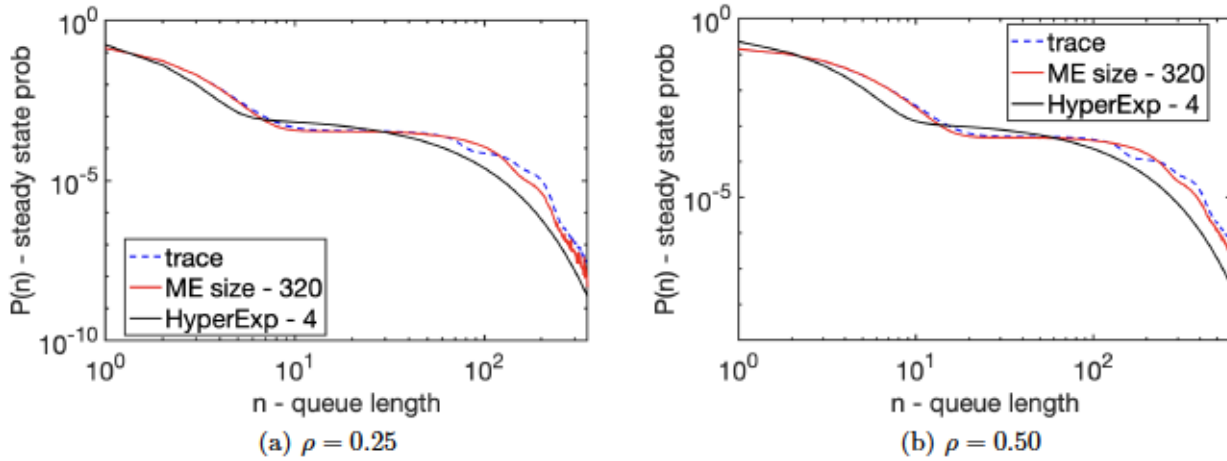


Figure 5.13: Steady state queueing probabilities for BuildServer00 trace at 25% and 50% utilisation levels.

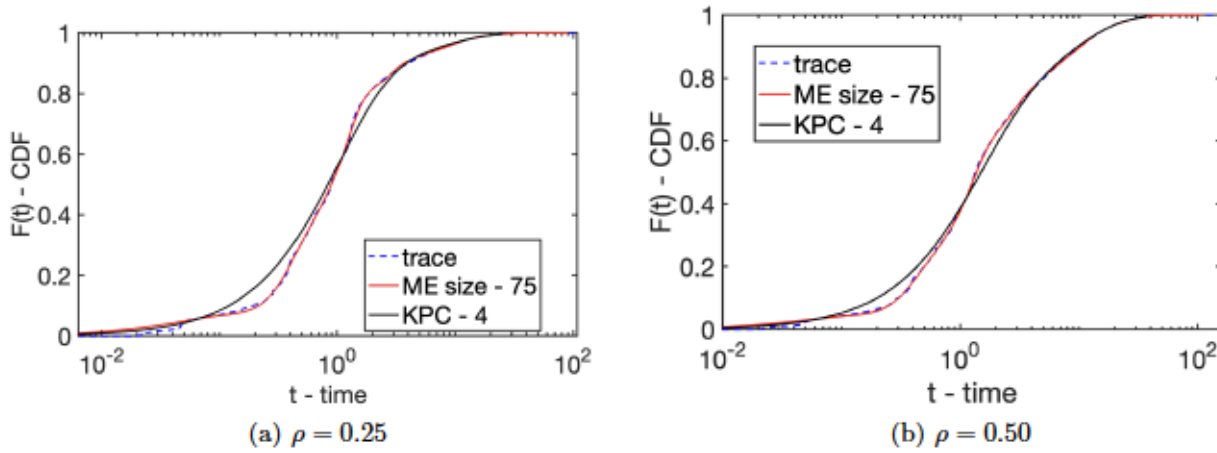


Figure 5.14: Response time CDF for BCAug89 trace at 25% and 50% utilisation levels.

- We derive theoretical results for the efficient approximation of marginal queue-length probabilities for a QBD process with PH-distributed arrival times and compositional PH-distributed service times.
- We formulate an optimisation problem that utilises this approximation of queue-length probabilities in its objective function and aims to find a QBD process with a KPC-PH service distribution, that closely matches a given set of marginal queue-length probabilities.

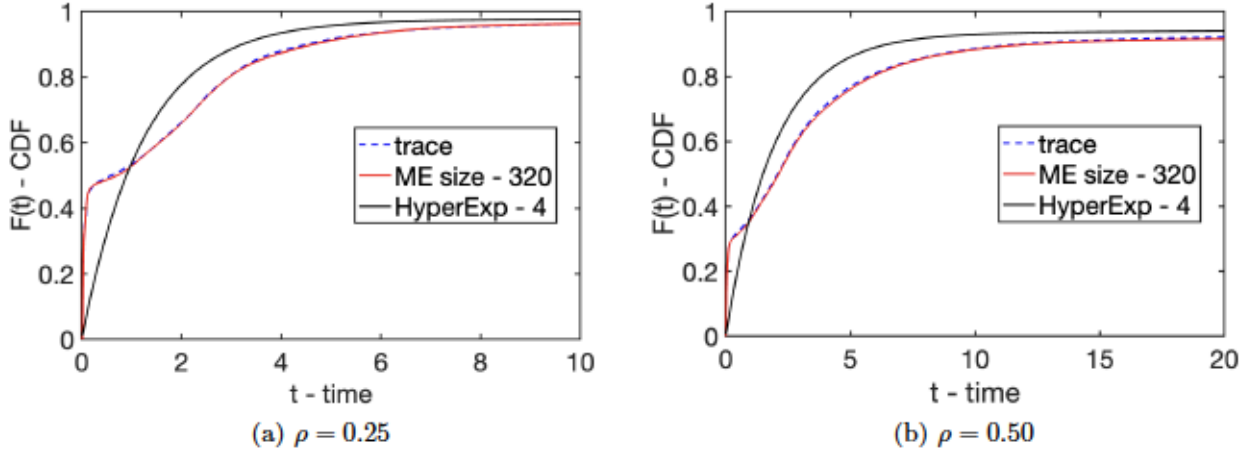


Figure 5.15: Response time CDF for BuildServer00 trace at 25% and 50% utilisation levels.

- We also develop an algorithm that reduces the size of these fitted distributions. When the large size is a symptom of overparameterization, this reduction maintains the accuracy of the distribution while eliminating unnecessary parameters.
- We develop a novel ME fitting algorithm based on a special family of ME distributions, constructed as a mixture of ME distributions. This approach allows for efficient calculation of the CDF and Mellin transform, which are then used to drive the fitting process.
- Numerical experiments show that both fitting algorithms perform extremely well compared to benchmark fitting algorithms both when comparing the CDFs of the fitted distributions to the empirical CDFs directly, and when comparing the steady-state queue-length distributions and response time distributions of a queue with the fitted distributions as its service distribution.

Chapter 6

Conclusion

In this chapter, we summarise the key achievements of the thesis, highlighting the insights gained and the broader contributions of the work. The chapter concludes by outlining potential future research directions based on the contributions of each chapter.

6.1 Summary of thesis achievements

This dissertation focuses on obtaining response time distributions in various queueing models by applying purely analytical approaches, approximation methods and their combinations. We derive two novel results for obtaining response time distributions through exact equations, which are numerically solved in a network of queues operating under the PS discipline. These results are derived under exponential inter-arrival and service time assumptions. We then examine queues with PH inter-arrival and service distributions, finding parameterisations for the service distribution that match an input trace accurately by approximately solving the corresponding QBD processes. Finally, we develop a novel algorithm that parametrises an ME distribution to match the statistical properties of a given input trace.

Consequently, this dissertation contributes both novel analytical insights and reusable numerical frameworks, which are summarised below for each chapter. By systematically developing and applying these two perspectives, the work demonstrates that analytical and numerical methods

are equally valuable, not competing alternatives, but complementary approaches for addressing the same class of queueing problems from different angles. This observation is important, as the complementarity between analytical and numerical approaches in queueing theory is often overlooked.

In Chapter 3, we address **RO1** by first considering a single M/M/1 queue under the PS discipline. We derive novel functional equations for this system, which are solved exactly. These functional equations are derived without conditioning on the service time of the tagged task, hence having one less variable transforming the PDE of [CJMT70] into an ODE. We show that the obtained LST matches the known results exactly. Although the resulting distribution itself is not new, the reformulation of the functional equation and the resulting simplification of the LST provides a more tractable analytical foundation, enabling its direct reuse in more complex queueing networks where the M/M/1-PS queue serves as a fundamental building block. Furthermore, in contrast with the LST conditioned on the tagged task service time, our solution is straightforwardly invertible using the Fixed Talbot inversion algorithm, requiring no extra effort to obtain the distribution itself. Next, we solve the obtained functional equations by applying the numerical approximation algorithm of [Har24] and perform an extensive evaluation of the method by comparing its performance to exact solutions. As discussed in Chapter 2, a key limitation of existing methods for obtaining response time distributions is their lack of reusability. Because the approximation method used here is highly adaptable, a thorough evaluation is especially valuable, as it demonstrates both the reliability of the method and its applicability to solving functional equations encountered in queueing problems.

Finally, we consider a tandem network of two PS queues and derive transformed versions of the functional equations given in [MM83] to enhance the accuracy of the numerical approximation method when solving these equations. Consequently, the response time distribution of a tandem pair of PS queues is derived for the first time, together with an efficient method of calculating the moments of the distribution. The results are compared to regenerative simulation and show very high accuracy. In Chapter 2, it is noted that although PS is not the most optimal scheduling algorithm, it is widely used in practice because it requires knowledge only of the queue lengths rather than the sizes of individual jobs at arrival or during their processing.

Furthermore, it is considered a fair scheduling policy, as it yields a constant mean slowdown independent of job sizes. Obtaining the response time distribution and moments for a tandem PS network allows us to quantify the additional variability introduced compared to a FCFS schedule in the context of tandem networks. Additionally, we gain the surprising insight that the PS-PS tandem network is likely insensitive to the order of the queues, even though the response times at the two queues are not independent.

In Chapter 4, we address **RO2** by obtaining the response time distribution in a pair of PS queues under JSQ scheduling, for the first time. We start by obtaining the functional equations for the generating functions of conditional response time LSTs where the conditioning is on a tagged task joining the first or the second queue when the queue lengths are i and j , respectively. The deconditioning step is not straightforward in this model due to the inter-dependency of the two queues; however, the unconditional LST is successfully obtained via a carefully chosen complex integral that selects certain, infinitely many indices of the product of the generating functions of conditional LSTs and the generating function of queue-length probabilities obtained in [Har24]. The approximation method is then extended to handle additional operations present in the functional equations which are more complex than any other model previously solved by the method. Additionally, an efficient recursive method is obtained for calculating the moments of the response time. Results are again compared to regenerative simulation and show high accuracy. Furthermore, two optimisation methods are proposed and evaluated for increased accuracy.

This result is perhaps the most significant in the dissertation, both because of the complexity of the problem and the practical importance of JSQ scheduling, particularly when combined with PS scheduling at individual queues, widely used by network providers, cloud servers, and traffic management systems. Obtaining the response time distribution for this system allows us to compare JSQ scheduling with a static scheduling method, demonstrating how the load-balancing aspect of JSQ reduces response time variability. While JSQ is not the optimal scheduling policy for parallel PS queues in general, as discussed in Chapter 2, it shares the practical advantage of PS scheduling: it is near-optimal while requiring only knowledge of queue lengths, which explains its frequent use in practice. Moreover, numeric experiments

reveal that the optimal routing strategy is to send jobs to the faster queue whenever the queues are of equal length, aligning with the decision that would be made by the more optimal OPT-0 strategy most of the time, though not in every instance.

Although the networks studied in Chapters 3 and 4 are small, consisting of only two PS queues, obtaining the response time distributions and moments in these systems is a step toward calculating the response time distributions in larger and more realistic networks of PS queues. These results effectively add new tools to the existing toolkit, facilitating the study of increasingly complex PS queueing networks as encountered in real-world applications.

In Chapter 5, our focus shifts to queues with more general arrival and service processes, initially, those with PH arrival and service times. We consider queues with compositional PH service distributions and derive a method to solve the resulting QBD via a divide-and-conquer method. In the special case of exponential inter-arrival times, the divide-and-conquer approach calculates the marginal queue-length probabilities of the original QBD exactly. In the case of PH arrivals, the algorithm effectively functions as an aggregation method, solving the original QBD approximately in quadratic time, compared to the cubic time required for a direct solution. The computational gains are then utilised in a novel fitting algorithm that finds the parameters of a service distribution via the arrival trace and marginal queue-length distributions. Furthermore, the compositional approach allows for fitting PH distributions with even thousands of states. We also propose a reduction algorithm that reduces the size of the fitted distributions while maintaining a good level of accuracy of the fit. The reduction algorithm helps obtain distributions of a manageable size for analytical solvers while also mitigating potential overparameterization. This issue could arise from fitting too closely to the tail of the input trace, though it is unlikely due to the expected noise in the trace, which primarily affects the tail of the distribution. Numerical results show that our method fits real-life traces exceptionally well, outperforming all benchmark methods. The increased accuracy translates to better response time predictions. These results address the first part of **RO3**.

Numerous PH fitting algorithms exist in the literature. However, our method is fundamentally different from existing approaches, as the service process is fitted implicitly based on

queue-length traces and information on the arrival process, rather than direct measurements of the service distribution, which are often unavailable in practice. Empirical queue-length distributions have not been used previously due to the computational cost of calculating these probabilities in a QBD, which makes incorporating this information into the fitting process infeasible. The theoretical results presented in Chapter 5, however, enable the efficient calculation of approximate queue-length probabilities by leveraging a divide-and-conquer approach for a family of PH distributions, allowing this information to be used in the fitting algorithm for the first time.

Finally, we explore fitting with ME distributions addressing the second part of **RO3**. The QBD results derived for compositional PH service distributions no longer hold; however, we successfully alter the construction of the fitted distribution to produce a valid ME distribution. This is achieved by considering a special family of ME distributions defined as a product of a certain family of ME distributions and probabilistic scaling factors. The resulting distribution is a mixture of ME distribution, for which the calculation of the CDF and Mellin transform can be decomposed and therefore obtained very efficiently. We show that a fitting algorithm utilising this special family of ME distributions that uses the CDF and Mellin transforms in its objective function performs well on real-life traces. We further validate the results by calculating marginal queue-length probabilities in queues with a fixed PH arrival and the fitted distribution as the service distribution and show that increased accuracy of the fitted distribution results in more accurate queue-length probabilities when compared to benchmark methods.

Using ME distributions for fitting retains the advantage of PH distributions, namely that any distribution over the positive reals can be approximated arbitrarily well by a PH distribution, while also removing non-negativity constraints on the parameters, which simplifies the optimisation problem. The main reason ME distributions are rarely used for fitting is that verifying the validity of a distribution for a given set of parameters is computationally expensive. Consequently, the key contribution here is the development of a numerical framework for fitting ME distributions that addresses this challenge by ensuring validity by construction. Furthermore, the method uses a compositional approach, enabling the fitting of potentially large ME distri-

butions and hence providing greater flexibility in the resulting fit. It also leverages the CDF in the fitting process, avoiding reliance on often inaccurate higher empirical moments.

It is also worth discussing the limitations of the work presented in the thesis. First, we consider only queues in steady state, whereas in practice this is not always the case. While steady-state results provide valuable benchmarks even when queues have not yet reached equilibrium, a direct analysis of transient queues — both for calculating response times and for fitting queues in transient states — would complement the current results. Second, our work focuses exclusively on unbounded queues, where new arrivals are always accepted. Bounded queues differ fundamentally from unbounded queues because arrivals can be rejected when the system reaches capacity. This introduces blocking events and additional state dependencies, which complicate both the queue-length and response time distributions. Consequently, analytical and numerical methods developed for unbounded queues usually cannot be applied directly, and instead, new approaches are required to handle the effects of finite capacity.

6.2 Future work

A natural future direction for the results obtained in Chapter 3 would be to consider more than two PS queues in tandem. Obtaining exact functional equations for more than two queues quickly becomes impractical as the number of queues increases. However, it might be possible to derive approximate solutions by considering two queues at the time assuming Poisson arrivals elsewhere and applying a reversed process argument similar to [HB21b]. That is, in the case of three queues, $t = 0$ is shifted to the time instant when the tagged task departs from queue 1. The joint distribution is then considered for the reversed response time at the first queue and the response times at the remaining two queues in isolation.

In the case of the JSQ model in Chapter 4, a natural extension is again to consider more than two queues. This model is significantly more complex than the tandem queue network, and even if exact functional equations are derived, the dimension of the matrices required for the approximation method to work accurately becomes too large, even for just three queues. However, a possible approach could be to split the number of queues into two groups at the

time and obtain an approximate solution via a divide-and-conquer approach. Alternatively, a fluid approximation could also be explored for deriving approximate results for a large number of parallel queues. Furthermore, it would be interesting to explore variations on JSQ scheduling prompted by the experiments in Section 4.4.3 corresponding to finding the optimal values of a_1 and the offset parameter k . These variations can be thought of as shifted versions of the original JSQ scheduling policy, where the faster server automatically receives new arrivals until its queue is k customers longer than the queue of the slower server. In addition to calculating the response time density for a shifted JSQ system, it is also interesting to try and find the optimal k value for a given set of service rates.

Regarding the PH fitting algorithm proposed in Chapter 5, future work could examine the extension of the presented results to a broader class of Markovian processes, such as Markovian arrival processes which can describe potential auto-correlation of the input traces. Queueing systems associated with MAPs can also be studied through matrix analytic methods. Therefore, our results may also be used to fit and analyse these processes. A similar extension could be explored for the ME fitting algorithm, where the corresponding process that takes auto-correlation into account is called a Rational Arrival Process. These extensions are worth considering, as many real-life traces exhibit correlation. Accounting for this would further broaden the applicability of our proposed algorithms.

Bibliography

- [ACC⁺14] Danilo Ardagna, Giuliano Casale, Michele Ciavotta, Juan F. Pérez, and Weikun Wang. Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, 5, 2014.
- [ACCP12] Danilo Ardagna, Sara Casolari, Michele Colajanni, and Barbara Panicucci. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. *Journal of Parallel and Distributed Computing*, 72, 2012.
- [AER⁺19] Ehsan Ataie, Reza Entezari-Maleki, Leila Rashidi, Kishor S. Trivedi, Danilo Ardagna, and Ali Movaghar. Hierarchical stochastic models for performance, availability, and power consumption analysis of IaaS clouds. *IEEE Trans. Cloud Computing*, 7(4):1039–1056, 2019.
- [AF83] Benedict K. Asare and Frank G. Foster. Conditional response times in the M/G/1 processor-sharing system. *Journal of Applied Probability*, 20(4), 1983.
- [Alp21] Ethem Alpaydin. *Machine Learning, revised and updated edition*. The MIT Press Essential Knowledge series. MIT Press, 2021.
- [AN98] Allan T Andersen and Bo Friis Nielsen. A Markovian approach for modeling packet traffic with long-range dependence. *IEEE JSAC*, 16(5):719–732, 1998.
- [ANO96] Søren Asmussen, Olle Nerman, and Marita Olsson. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 1996.
- [ANT21] Azam Asanjarani, Yoni Nazarathy, and Peter Taylor. A survey of parameter and state estimation in queues. *Queueing Systems*, 97:39–80, 2021.

- [APYS22] Ahsan Ali, Riccardo Pincioli, Feng Yan, and Evgenia Smirni. Optimizing inference serving on serverless platforms. *Proceedings of the VLDB Endowment*, 15(10), 2022.
- [AS87] David Aldous and Larry Shepp. The least variable phase type distribution is Erlang. *Stochastic Models*, 3(3):467–473, 1987.
- [AW95] Joseph Abate and Ward Whitt. Numerical inversion of Laplace transforms of probability distributions. *ORSA Journal on Computing*, 7(1):36–43, 1995.
- [BB06] Lothar Breuer and Dieter Baum. *An Introduction to Queueing Theory and Matrix-Analytic Methods*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [BC09] James Ward Brown and Ruel V Churchill. *Complex variables and applications*. McGraw-Hill,, 2009.
- [BC17] Abhijit Datta Banik and M. L. Chaudhry. Efficient computational analysis of stationary probabilities for the queueing system $BMAP/G/1/N$ with or without vacation(s). *INFORMS Journal on Computing*, 29(1):140–151, 2017.
- [BCK⁺23] Julianna Bor, Giuliano Casale, William Knottenbelt, Evgenia Smirni, and Andreas Stathopoulos. Fitting with matrix exponential mixtures generated by discrete probabilistic scaling. *SIGMETRICS Perform. Eval. Rev.*, 51(2):15–17, October 2023.
- [BCS25] Julianna Bor, Giuliano Casale, and Evgenia Smirni. The component queues method for quasi-birth-death process fitting and analysis. unpublished, 2025.
- [BEN11] Mogens Bladt, Luz Judith R. Esparza, and Bo Friis Nielsen. Fisher information and statistical inference for phase-type distributions. *Journal of Applied Probability*, 48(A):277–293, 2011.
- [BGDMT06] Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

- [BGN00] Richard Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 11 2000.
- [BH24] Julianna Bor and Peter G. Harrison. Response time in a pair of processor sharing queues with Join-the-Shortest-Queue scheduling. In *2024 32nd International Conference on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–8, 2024.
- [BH25a] Julianna Bor and Peter G. Harrison. Response time in a pair of processor sharing queues with Join-the-Shortest-Queue scheduling. *Performance Evaluation*, 170:102509, 2025.
- [BH25b] Julianna Bor and Peter G. Harrison. Response time in a tandem network of two processor sharing queues. In *Proceedings of the 21st European Performance Engineering Workshop*, 2025.
- [BHB01] Nikhil Bansal and Mor Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. In *Proceedings of the 2001 ACM SIGMETRICS International conference on Measurement and modeling of computer systems*, pages 279–290, 2001.
- [BHN99] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.
- [BHT05] Andrea Bobbio, András Horváth, and Miklós Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic models*, 21(2-3):303–326, 2005.
- [BHT11] Peter Buchholz, A HORVÁTHand, and Miklos Telek. Stochastic Petri nets with low variation matrix exponentially distributed firing time. *International Journal of Performability Engineering*, 7:441–454, 2011.

- [BKK84] Onno J Boxma, Frank P Kelly, and Alan G Konheim. The product form for sojourn time distributions in cyclic exponential queues. *Journal of the ACM*, 31(1):128–133, January 1984.
- [Bla05] Mogens Bladt. A review on phase-type distributions and their use in risk theory. *ASTIN Bulletin: The Journal of the IAA*, 35(1):145–161, 2005.
- [BN12] Nigel G. Bean and Bo Friis Nielsen. Analysis of queues with rational arrival process components: A general approach. *ACM SIGMETRICS Performance Evaluation Review*, 39(4):31, 2012.
- [BN17] Mogens Bladt and Bo Friis Nielsen. *Matrix-Exponential Distributions in Applied Probability*, volume 81. Springer New York, NY, 2017.
- [BNN16] Mogens Bladt, Azucena Campillo Navarro, and Bo Friis Nielsen. On the use of functional calculus for phase-type and related distributions. *Stochastic Models*, 32(1):1–19, 2016.
- [Bon02] Flavio Bonomi. On job assignment for a parallel system of processor sharing queues. *IEEE Transactions on Computers*, 39(7):858–869, 2002.
- [BR05] James Baglama and Lothar Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 27(1):19–42, 2005.
- [Buc94] Peter Buchholz. *On a Markovian process algebra*. Citeseer, 1994.
- [BWB⁺19] Liang Bao, Chase Wu, Xiaoxuan Bu, Nana Ren, and Mengqing Shen. Performance modeling and workflow scheduling of microservice-based applications in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 30(9):2114–2129, 2019.
- [CC90] Knessl Charles and Tier Charles. Approximations to the moments of the sojourn time in a tandem queue with overtaking. *Stochastic Models*, 6(3):499–524, 1990.
- [Chi95] Chiahon Chien. Batch size selection for the batch means method. In *Proceedings of the 1994 Winter Sim. Conf.*, pages 345– 352, 01 1995.

- [CJFM86] Edward G Coffman Jr, Guy Fayolle, and Isi Mitrani. Sojourn times in a tandem queue with overtaking: reduction to a boundary value problem. *Stochastic Models*, 2(1):43–65, 1986.
- [CJMT70] Edward G Coffman Jr, Richard R Muntz, and Hale Trotter. Waiting time distributions for processor-sharing systems. *Journal of the ACM (JACM)*, 17(1):123–130, 1970.
- [CKT12] Yung-Chih Chen, Jim Kurose, and Don Towsley. A mixed queueing network model of mobility in a campus wireless network. In *2012 Proceedings IEEE INFOCOM*, pages 2656–2660. IEEE, 2012.
- [CRS01] Gianfranco Ciardo, Alma Riska, and Evgenia Smirni. Equiloader: a load balancing policy for clustered web servers. *PEVA*, 46(2-3), 2001.
- [CS12] Giuliano Casale and Evgenia Smirni. KPC-toolbox: fitting markovian arrival processes and phase-type distributions with matlab. *ACM SIGMETRICS Performance Evaluation Review*, 39(4):47, apr 2012.
- [CZS10a] Giuliano Casale, Eddy Z Zhang, and Evgenia Smirni. KPC-toolbox: Best recipes for automatic trace fitting using Markovian Arrival Processes. *Performance Evaluation*, 67(9):873–896, 2010.
- [CZS10b] Giuliano Casale, Eddy Z. Zhang, and Evgenia Smirni. Trace data characterization and fitting for Markov modeling. *Performance Evaluation*, 67(2):61–79, 2010.
- [EM17] Paul D. Ezhilchelvan and Isi Mitrani. Evaluating the probability of malicious co-residency in public clouds. *IEEE Transactions on Cloud Computing*, 5(3):420–427, July 2017.
- [Fac05] Mark Fackrell. Fitting with matrix-exponential distributions. *Stochastic Models*, 21(2-3):377–400, 2005.
- [FM77] Leonard Flatto and Henry P. McKean. Two queues in parallel. *Communications on Pure and Applied Mathematics*, 30(2):255–263, 1977.

- [FW98] Anja Feldmann and Ward Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance evaluation*, 31(3-4):245–279, 1998.
- [GAGV15] Eduardo Gonzalo-Almorox, Alexandru Lucian Gaianu, and Luke Vale. The role of queuing models in economic evaluation: lessons from an illustrative case study of emergency care, 2015.
- [GBSW07] Varun Gupta, Mor Harchol Balter, Karl Sigman, and Ward Whitt. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation*, 64(9-12):1062–1081, 2007.
- [Gly06] Peter W Glynn. Simulation algorithms for regenerative processes. *Handbooks in Operations Research and Management Science*, 13:477–500, 2006.
- [GOST15] Fabrice Guillemin, Patrick Olivier, Aram Simonian, and Claude Tanguy. Two parallel queues with infinite servers and join the shortest queue discipline. *Stochastic Models*, 31(4):636–672, 2015.
- [GRS18] Fabrice Guillemin, Veronika Karina Quintuna Rodriguez, and Alain Simonian. Sojourn time in a processor sharing queue with batch arrivals. *Stochastic Models*, 34, 2018.
- [GRS19] Fabrice Guillemin, Veronica Quintuna Rodriguez, and Alain Simonian. A processor-sharing model for the performance of virtualized network functions. In *2019 31st International Teletraffic Congress (ITC 31)*, pages 10–18. IEEE, 2019.
- [GW19] Erol Gelenbe and Yi Wang. Supply chains for perishable goods and G-networks. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 167–172. IEEE, 2019.
- [Har08] Harro L. Hartmann. The uniform estimation of the M/G/1 processor sharing response time distribution. *European Transactions on Telecommunications*, 19:619–633, 10 2008.

- [Har24] Peter G. Harrison. On the numerical solution of functional equations with application to response time distributions. *Applied Mathematics and Computation*, 472, 2024.
- [HB00] Mor Harchol-Balter. Task assignment with unknown duration. In *Proceedings 20th IEEE international conference on distributed computing systems*, pages 214–224. IEEE, 2000.
- [HB21a] Peter G. Harrison and Julianna Bor. Response time distribution in a tandem pair of queues with batch processing. *Journal of the ACM*, 68, 2021.
- [HB21b] Peter G. Harrison and Julianna Bor. Response time distribution in a tandem pair of queues with batch processing. *Journal of the ACM*, 68(4), June 2021.
- [HBCM99] Mor Harchol-Balter, Mark E Crovella, and Cristina D Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59(2):204–228, 1999.
- [HKH⁺16] Hourvash Akbari Haghighinejad, Erfan Kharazmi, Nahid Hatam, Sedigheh Yousefi, Seyed Ali Hesami, Mina Danaei, and Mehrdad Askarian. Using queuing theory and simulation modelling to reduce waiting times in an Iranian emergency department. *International journal of community based nursing and midwifery*, 4(1):11, 2016.
- [HM98] Carl M. Harris and William G. Marchal. Distribution estimation using Laplace transforms. *INFORMS journal on Computing*, 10:448–458, 1998.
- [Hor73] Lars Hormander. *An Introduction to Complex Analysis in Several Variables*. Elsevier, 1973.
- [Hor13] Gábor Horváth. Moment matching-based distribution fitting with generalized hyper-Erlang distributions. In *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pages 232–246. Springer, 2013.

- [HP92] Peter G. Harrison and Naresh M. Patel. *Performance Modelling of Communication Networks and Computer Architectures (International Computer S)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1992.
- [HP93] Peter G Harrison and Edwige Pitel. Sojourn times in single-server queues with negative customers. *Journal of Applied Probability*, 30(4):943–963, 1993.
- [HP95] Peter G. Harrison and Edwige Pitel. Response time distributions in tandem G-networks. *Journal of Applied Probability*, 32(1):224–246, 1995.
- [HP18] Peter G. Harrison and Naresh M Patel. Optimizing energy-performance trade-offs in solar-powered edge devices. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 253–260, 2018.
- [HQ13] Peter G. Harrison and Zhan Qiu. Performance enhancement by means of task replication. In *European Workshop on Performance Engineering*, pages 191–205. Springer, 2013.
- [HSTZ16] Illés Horváth, Orsolya Sáfár, Miklós Telek, and Bence Zámbo. Concentrated matrix exponential distributions. In *European Workshop on Performance Engineering*, pages 18–31. Springer, 2016.
- [HT02] András Horváth and Miklós Telek. Markovian modeling of real data traffic: Heuristic phase type and MAP fitting of heavy tailed and fractal like samples. In *IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation*, pages 405–434. Springer, 2002.
- [HT16] Gábor Horváth and Miklós Telek. Butools 2: a rich toolbox for Markovian performance evaluation. In *Valuetools*. ACM, 2016.
- [HV23] András Horváth and Enrico Vicario. Construction of phase type distributions by Bernstein exponentials. In *Computer Performance Engineering and Stochastic Modelling*, pages 201–215, Cham, 2023. Springer Nature Switzerland.

- [Kin61] John FC Kingman. Two Similar Queues in Parallel. *The Annals of Mathematical Statistics*, 32(4):1314 – 1323, 1961.
- [Kle67] Leonard Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the ACM (JACM)*, 14, 1967.
- [KML05] Jayesh Kumaran, Ken Mitchell, and A Liefvoort. The waiting time distribution of an MEP/MEP/1 queue. In *Proceedings of the 19th International Teletraffic Congress (ITC19)*, pages 687–696, 2005.
- [Kne93] Charles Knessl. On the sojourn time distribution in a finite capacity processor shared queue. *Journal of the ACM (JACM)*, 40, 1993.
- [KRCW75] Peter J. Kolesar, Kenneth L. Rider, Thomas B. Crabill, and Warren E. Walker. A queuing-linear programming approach to scheduling police patrol cars. *Operations Research*, 23(6):1045–1062, 1975.
- [LR99] Guy Latouche and Vaidyanathan Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*. SIAM, 1999.
- [LS04] Amy N. Langville and William J. Stewart. The Kronecker product and stochastic automata networks. *Journal of Computational and Applied Mathematics*, 167(2):429–447, 2004.
- [LW13] Wei Yi Lee and Chia Li Wang. Conditional sojourn times of processor-sharing queues. *Probability in the Engineering and Informational Sciences*, 27:99–114, 2013.
- [Mat24] MathWorks. fmincon documentation home. <https://uk.mathworks.com/help/optim/ug/fmincon.html>, 2024.
- [MM83] Debasis Mitra and J. A. Morrison. Asymptotic expansions of moments of the waiting time in closed and open processor-sharing systems with multiple job classes. *Advances in Applied Probability*, 15, 1983.

- [MM85] J. A. Morrison and D. Mitra. Heavy-usage asymptotic expansions for the waiting time in closed processor-sharing systems with multiple classes. *Advances in Applied Probability*, 17, 1985.
- [MO92] Robert S. Maier and Colm Art O’Cinneide. A closure characterisation of phase-type distributions. *Journal of Applied Probability*, 29(1):92–103, 1992.
- [Mor85] John A. Morrison. Response-time distribution for a processor-sharing system. *SIAM Journal on Applied Mathematics*, 45(1):152–167, 1985.
- [Neu82] Marcel F Neuts. Explicit steady-state solutions to some elementary queueing models. *Operations Research*, 30(3):480–489, 1982.
- [Neu94] Marcel F Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Courier Corporation, 1994.
- [NMM98] Masahiko Nabe, Masayuki Murata, and Hideo Miyahara. Analysis and modeling of World Wide Web traffic for capacity dimensioning of internet access lines. *Performance Evaluation*, 34:249–271, 12 1998.
- [NT13] Walid W. Nasr and Michael R. Taaffe. Fitting the $Ph_t/M_t/s/c$ time-dependent departure process for use in tandem queueing networks. *INFORMS Journal on Computing*, 25(4):758–773, 2013.
- [ODT13] Hiroyuki Okamura, Tadashi Dohi, and Kishor S Trivedi. Improvement of expectation–maximization algorithm for phase-type distributions with grouped and truncated data. *Applied Stochastic Models in Business and Industry*, 29(2):141–156, 2013.
- [Pri03] Hilary A Priestley. *Introduction to Complex Analysis second edition (2nd ed.)*. Oxford University Press, 2003.
- [PS03] Athanasios E. Papathanasiou and Michael L. Scott. Energy efficiency through burstiness. In *2003 Proceedings Fifth IEEE Workshop on Mobile Computing Systems and Applications*, pages 44–53, 2003.

- [PSCS⁺11] Sergio Pacheco-Sanchez, Giuliano Casale, Bryan Scotney, Sally McClean, Gerard Parr, and Stephen Dawson. Markovian workload characterization for QoS prediction in the cloud. In *IEEE CLOUD 2011*, pages 147–154. IEEE, 2011.
- [PVVVH08] Juan F Pérez, Jeroen Van Velthoven, and Benny Van Houdt. Q-MAM: A tool for solving infinite queues using matrix-analytic methods. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*, page 9, Brussels, BEL, 2008.
- [RDS04] Alma Riska, Vesselin Diev, and Evgenia Smirni. An EM-based technique for approximating long-tailed data sets with PH distributions. *Performance Evaluation*, 55(1):147–164, 2004.
- [RH12] Philipp Reinecke and Gábor Horváth. Phase-type distributions for realistic modelling in discrete-event simulation. In *SimuTools*, pages 283–290, 2012.
- [RIS22] Amir Rastpour, Armann Ingolfsson, and Burhaneddin Sandikçi. Algorithms for queueing systems with reneging and priorities modeled as quasi-birth-death processes. *INFORMS Journal on Computing*, 34(3):1693–1710, 2022.
- [RPT⁺15] Jacek Rak, Mario Pickavet, Kishor S Trivedi, Javier Alonso Lopez, Arie MCA Koster, James PG Sterbenz, Egemen K Çetinkaya, Teresa Gomes, Matthias Gunkel, Krzysztof Walkowiak, et al. Future research directions in design of reliable communication systems. *Telecommunication Systems*, 60:423–450, 2015.
- [RS02] Alma Riska and Evgenia Smirni. M/G/1-type Markov processes: A tutorial. In *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, volume 2459 of *Lecture Notes in Computer Science*, pages 36–63. Springer, 2002.
- [RS07] Alma Riska and Evgenia Smirni. ETAQA solutions for infinite markov processes with repetitive structure. *IJOC*, 19(2), 2007.
- [Sch68] Linus Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.

- [Smi78] Donald R Smith. A new proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 26(1):197–199, 1978.
- [TBT06] Axel Thümmler, Peter Buchholz, and Miklós Telek. A novel approach for PH fitting with the EM algorithm. *IEEE Transactions on dependable and secure computing*, 3(3):245–258, 2006.
- [TGL24] Kishor S. Trivedi, Michael Grottke, and Javier Alonso Lopez. Rethinking software fault tolerance. *IEEE Transactions on Reliability*, 73(1):67–72, 2024.
- [Vis03] Vladimir M. Vishnevskii. Teoreticheskie osnovy proektirovaniya komp'yuternykh setei (Theoretical Fundamentals of Computer Network Design). *M: Tekhnosfera*, 2003.
- [WA19] Application load balancer now supports least outstanding requests algorithm for load balancing requests. <https://shorturl.at/ZEGw9>, 2019. Accessed on March 3, 2025.
- [WAZ23] Microsoft.network applicationgateways. <https://learn.microsoft.com/en-us/azure/templates/microsoft.network/applicationgateways?pivots=deployment-language-bicep>, 2023. Accessed on March 3, 2025.
- [WC11] Cisco IOS server load balancing configuration guide. https://www.cisco.com/c/en/us/td/docs/ios/slb/configuration/guide/slb_cg_book/slb_cg_info.html, 2011. Accessed on March 3, 2025.
- [Web78] Richard R Weber. On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15(2):406–413, 1978.
- [Win77] Wayne Winston. Optimality of the shortest line discipline. *Journal of applied probability*, 14(1):181–189, 1977.
- [WK23] Virtual IPs and service proxies. <https://kubernetes.io/docs/reference/networking/virtual-ips/>, 2023. Accessed on March 3, 2025.

- [XWM⁺11] Pengcheng Xiong, Zhikui Wang, Simon Malkowski, Qingyang Wang, Deepal Jayasinghe, and Calton Pu. Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach. In *2011 31st International Conference on Distributed Computing Systems*, pages 571–580. IEEE, 2011.
- [YHB12] Kaiqi Yu, Mei-Ling Huang, and Percy H. Brill. An algorithm for fitting heavy-tailed distributions via generalized hyperexponentials. *INFORMS Journal on Computing*, 24(1):42–52, 2012.
- [ZK09a] Qiang Zhen and Charles Knessl. On sojourn times in the finite capacity M/M/1 queue with processor sharing. *Operations Research Letters*, 37:447–450, 11 2009.
- [ZK09b] Qiang Zhen and Charles Knessl. On sojourn times in the M/M/1-PS model, conditioned on the number of other users. *Applied Mathematics Research eXpress*, 2009, 2009.

Appendices

A.1 Chapter 3

A.1.1 Proofs

Proof of Lemma 3.2.1.

$$\left| \int_{\gamma} f_n - \int_{\gamma} f \right| \leq \int_{\gamma} |f_n - f| \leq \epsilon l(\gamma)$$

where $l(\gamma)$ is the length of the curve and $\exists N : |f_n(t) - f(t)| \leq \epsilon \forall \gamma(t)$ when $n > N$ due to the uniform convergence condition. $\square$

Proof of Lemma 3.2.2. Using M_f and M_g to denote the bounds for f and g respectively,

$$\begin{aligned} |f_n g_n - f g| &= |f_n g_n - f g_n + f g_n - f g| = |(f_n - f)g_n + f(g_n - g)| \leq \\ &\leq |f_n - f||g_n| + |(g_n - g)||f| \leq \epsilon(2M_g + M_f) \end{aligned}$$

where for the last step we use that

- $\exists N_1 : n > N_1 : |f_n(t) - f(t)| < \epsilon \forall t$
- $\exists N_2 : n > N_2 : |g_n(t) - g(t)| < \epsilon \forall t$
- $\exists N_3 : n > N_3 : |g_n(t)| < 2M_g \forall t$

Hence whenever $n \geq \max(N_1, N_2, N_3)$ the inequality holds for all t -s in the domain, that is we have uniform convergence. $\square$

Proof of Lemma 3.2.3. Since B is a compact set, on B g is uniformly continuous due to the Heine-Cantor theorem, so that

$$|g(x) - g(y)| \leq \epsilon \text{ if } |x - y| \leq \delta \ \forall x, y \in B$$

Furthermore, $f_n \rightarrow f$ uniformly, which means:

$$\exists N_f : n > N : |f(t) - f_n(t)| \leq \delta \ \forall t$$

Picking the threshold to be $N := \max(N_B, N_f)$ for $n > N$ we have:

$$|g(f(t)) - g(f_n(t))| \leq \epsilon \ \forall t$$

$\square$

Proof of Lemma 3.2.4. $B_1 \times B_2$ is a finite product of compact sets, therefore it is itself compact. Consequently, it is enough to prove continuity of F and uniform continuity will follow.

$$|x_1^{y_1} - x_0^{y_0}| = |x_1^{y_1} - x_1^{y_0} + x_1^{y_0} - x_0^{y_0}| \leq |x_1^{y_1} - x_1^{y_0}| + |x_1^{y_0} - x_0^{y_0}|$$

Considering the terms one by one:

1. $\forall x_1 \in B_1 \setminus \{0\} : |x_1^{y_1} - x_1^{y_0}| < \epsilon/2$ if $y_0, y_1 \in B_2$ are such that $|y_1 - y_0| < \delta_{x_1, \epsilon}$. Furthermore, when $x_1 = 0$, $x_1^y = 0^y$ can be defined as $\equiv 0$, since $\Re(y) > 0$ on B_2 .
2. $y_0 \in B_2 : |x_1^{y_0} - x_0^{y_0}| < \epsilon/2$ if $x_0, x_1 \in B_1$ are such that $|x_1 - x_0| < \delta_{y_0, \epsilon}$.

Setting $\delta_{x_1, y_0, \epsilon} := \min(\delta_{x_1, \epsilon}, \delta_{y_0, \epsilon})$, and requiring $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} < \delta_{x_1, y_0, \epsilon}$ ensures both $|y_1 - y_0| < \delta_{y_0}$ and $|x_1 - x_0| < \delta_{x_1}$. Therefore, we get $|x_1^{y_1} - x_1^{y_0}| + |x_1^{y_0} - x_0^{y_0}| < \epsilon/2 + \epsilon/2 = \epsilon$ as required to prove continuity. $\square$

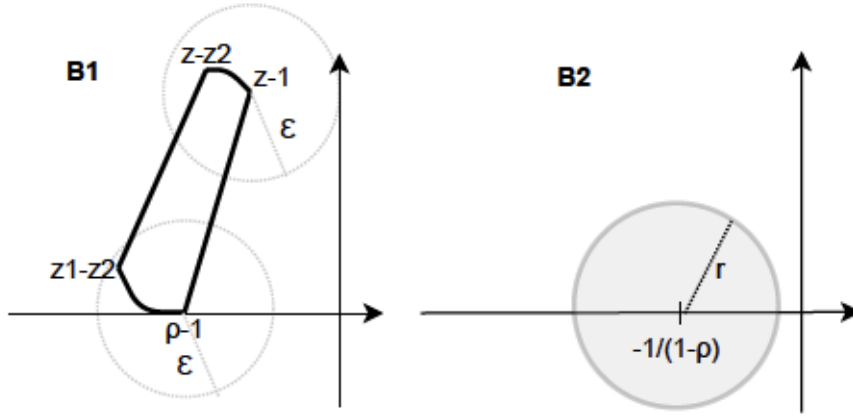


Figure 1: Compact sets $B1 : \text{Im}(f_n) \subseteq B1$ and $B2 : \text{Im}(g_n) \subseteq B2$

Proof of Part 3 in Proposition 3.2.3. Using the notations of Lemma 3.2.4, we have

$$f_n(x) := \frac{z - z_1^{(n)}}{z} x + z_1^{(n)} - z_2^{(n)}, \quad g_n(x) := \frac{-z_2^{(n)}}{z_2^{(n)} - z_1^{(n)}} \text{ so that } r_n(x) = f_n(x)^{g_n(x)}$$

$$f(x) := \frac{z - \rho}{z} x + \rho - 1, \quad g(x) := \frac{-1}{1 - \rho} \text{ so that } r(x) = f(x)^{g(x)}$$

Let $B1$ and $B2$ be defined as

$$B1 := \{f_n(x) | x \in [0, z]\} \cup \{f(x) | x \in [0, z]\}$$

$$B2 := \overline{B} \left(\frac{-1}{1 - \rho}, r \right), \quad r < \frac{-1}{1 - \rho}$$

The sets are shown in figure 1. Note that $0 \notin B1$ this time since $\rho - 1 < 0$ and $\Re(z - 1) < 0$. Furthermore, $\Re(z_1 - z_2) < 0 \Leftrightarrow \Re(z_1) < \Re(z_2)$ and $\Re(z - z_2) < 0 \Leftrightarrow \Re(z) < \Re(z_2)$ both of which are true if $|s|$ is small as $z_1 \rightarrow \rho$ and $z_2 \rightarrow 1$ when $s \rightarrow 0$. From here, the reasoning is analogous to Part 2. $\square$

A.1.2 Tables and figures

Matrix operations for one variable

Copied from [Har24].

Operation	Matrix Representation
Identity operator giving G	$I^{n \times n}$
Multiplication by known function $A(z)$	$\text{Diag}(\vec{a})$
Differentiation d times wrt z	M_d
Value at internal point w	$K_w^{(0)}$

Table 1: Matrices representing operations in the summands of the functional equation for G : G itself; multiplication by a known fixed function $a(z)$; differentiation; and evaluation at a fixed internal point w .

Introductory example matrices

$$M_1 = \begin{pmatrix} -1.14 & 0.39 + 0.37i & 0.08 - 0.01i & 0.04 & 0.03 & 0.04 & 0.08 + 0.01i & 0.39 - 0.37i \\ 0.01 - 0.54i & -0.81 + 0.81i & 0.54 - 0.01i & 0.05 - 0.06i & 0.03 - 0.03i & 0.02 - 0.02i & 0.03 - 0.03i & 0.06 - 0.05i \\ 0.01 - 0.08i & -0.37 - 0.39i & 0. + 1.14i & 0.37 - 0.39i & -0.01 - 0.08i & 0. - 0.04i & 0. - 0.03i & 0. - 0.04i \\ -0.03 - 0.03i & -0.05 - 0.06i & -0.54 - 0.01i & 0.81 + 0.81i & -0.01 - 0.54i & -0.06 - 0.05i & -0.03 - 0.03i & -0.02 - 0.02i \\ -0.03 & -0.04 & -0.08 - 0.01i & -0.39 + 0.37i & 1.14 & -0.39 - 0.37i & -0.08 + 0.01i & -0.04 \\ -0.03 + 0.03i & -0.02 + 0.02i & -0.03 + 0.03i & -0.06 + 0.05i & -0.01 + 0.54i & 0.81 - 0.81i & -0.54 + 0.01i & -0.05 + 0.06i \\ 0.01 + 0.08i & 0. + 0.04i & 0. + 0.03i & 0. + 0.04i & -0.01 + 0.08i & 0.37 + 0.39i & 0. - 1.14i & -0.37 + 0.39i \\ 0.01 + 0.54i & 0.06 + 0.05i & 0.03 + 0.03i & 0.02 + 0.02i & 0.03 + 0.03i & 0.05 + 0.06i & 0.54 + 0.01i & -0.81 - 0.81i \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} -0.97 - 0.67i & 0.94 - 0.94i & 1.25 + 0.64i & 0.01 + 1.37i & -1.06 + 0.67i & -0.93 - 0.43i & -0.08 - 0.64i & -0.08 - 0.21i \\ -0.64 + 0.08i & -0.73 - 1.56i & 1.25 - 0.64i & 0.82 + 0.82i & -0.32 + 0.89i & -0.7 + 0.12i & -0.27 - 0.32i & -0.07 - 0.13i \\ -0.32 + 0.27i & -0.93 - 0.43i & 0.31 - 1.94i & 1.37 + 0.01i & 0.32 + 0.89i & -0.44 + 0.44i & -0.31 - 0.09i & -0.04 - 0.09i \\ -0.09 + 0.31i & -0.7 + 0.12i & -0.67 - 1.06i & 1.44 - 1.44i & 1.06 + 0.67i & -0.12 + 0.7i & -0.31 + 0.09i & 0. - 0.08i \\ 0.09 + 0.31i & -0.44 + 0.44i & -0.89 - 0.32i & -0.01 - 1.37i & 1.94 - 0.31i & 0.43 + 0.93i & -0.27 + 0.32i & 0.04 - 0.09i \\ 0.32 + 0.27i & -0.12 + 0.7i & -0.89 + 0.32i & -0.82 - 0.82i & 0.64 - 1.25i & 1.56 + 0.73i & -0.08 + 0.64i & 0.07 - 0.13i \\ 0.64 + 0.08i & 0.43 + 0.93i & -0.67 + 1.06i & -1.37 - 0.01i & -0.64 - 1.25i & 0.94 - 0.94i & 0.67 + 0.97i & 0.08 - 0.21i \\ 0.97 - 0.67i & 1.56 + 0.73i & 0.31 + 1.94i & -1.44 + 1.44i & -1.94 - 0.31i & -0.73 - 1.56i & 0.67 - 0.97i & 0. - 0.33i \end{pmatrix}$$

A.2 Chapter 5

A.2.1 Tables and figures

Table 2: Kolmogorov Smirnov distance for the CDF between the fitted distributions and the trace

(a) Radius Authentication		(b) Buildserver00	
<i>Fitted distribution</i>	<i>KS distance</i>	<i>Fitted distribution</i>	<i>KS distance</i>
KPC-QBD-960	0.2	KPC-QBD-1875	0.3
exact QBD-6	0.6	exact QBD-120	0.5
Prony's-2	0.6	Prony's-2	0.5
KPC-PH-8	0.8	KPC-PH-4	0.4
HyperExp-4	0.6	HyperExp-4	0.5

Table 3: KS distance for the response time CDF between the fitted distributions and the trace ($\rho = 0.25$)

(a) Radius Authentication		(b) Buildserver00	
<i>Fitted distribution</i>	<i>KS distance</i>	<i>Fitted distribution</i>	<i>KS distance</i>
KPC-QBD-960	0.1	KPC-QBD-1875	0.2
exact QBD-6	0.4	exact QBD-120	0.4
KPC-QBD-130 red	0.1	KPC-QBD-110 red	0.2
Prony's-2	0.4	Prony's-2	0.4
KPC-PH-8	0.6	KPC-PH-4	0.3
HyperExp-4	0.4	HyperExp-4	0.4

Table 4: KS distance for the response time CDF between the fitted distributions and the trace ($\rho = 0.5$)

(a) Radius Authentication		(b) Buildserver00	
<i>Fitted distribution</i>	<i>KS distance</i>	<i>Fitted distribution</i>	<i>KS distance</i>
KPC-QBD-960	0.1	KPC-QBD-1875	0.1
exact QBD-6	0.3	exact QBD-120	0.2
KPC-QBD-130 red	0.1	KPC-QBD-110 red	0.1
Prony's-2	0.3	Prony's-2	0.2
KPC-PH-8	0.4	KPC-PH-4	0.2
HyperExp-4	0.3	HyperExp-4	0.2

A.2.2 Fitted distribution — Radius Authentication trace

N = Erlang distribution with mean 1 and size 15

$$\begin{aligned}
 H_1 &= \begin{bmatrix} -7.9864 & 0 & 0 & 0 \\ 0 & -0.9573 & 0 & 0 \\ 0 & 0 & -108.0970 & 0 \\ 0 & 0 & 0 & -1.2930 \end{bmatrix}, \begin{bmatrix} 0.0467 & 6.8995 & 0.0413 & 0.9989 \\ 0.0056 & 0.8270 & 0.0049 & 0.1197 \\ 0.6318 & 93.3866 & 0.5587 & 13.5199 \\ 0.0076 & 1.1171 & 0.0067 & 0.1617 \end{bmatrix} \\
 H_2 &= \begin{bmatrix} -0.0050 & 0 & 0 & 0 \\ 0 & -0.0053 & 0 & 0 \\ 0 & 0 & -4.9672 & 0 \\ 0 & 0 & 0 & -0.1166 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0.0047 & 0.0003 \\ 0 & 0 & 0.0050 & 0.0003 \\ 0.0042 & 0.0048 & 4.6933 & 0.2650 \\ 0.0001 & 0.0001 & 0.1102 & 0.0062 \end{bmatrix} \\
 H_3 &= \begin{bmatrix} -1.5818 & 0 & 0 & 0 \\ 0 & -0.4281 & 0 & 0 \\ 0 & 0 & -0.1787 & 0 \\ 0 & 0 & 0 & -0.1891 \end{bmatrix}, \begin{bmatrix} 1.3809 & 0.1217 & 0.0189 & 0.0603 \\ 0.3737 & 0.0329 & 0.0051 & 0.0163 \\ 0.1560 & 0.0137 & 0.0021 & 0.0068 \\ 0.1651 & 0.0145 & 0.0023 & 0.0072 \end{bmatrix}
 \end{aligned}$$