University of London
Imperial College of Science, Technology and Medicine
Department of Computing

Steady-State and Response Time Analysis of Modulated Queues and Networks with Batches

Harf Zatschler

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the University of London and
the Diploma of Imperial College, October 2004

Abstract

Rapid growth in size and complexity of communication networks such as the Internet during the past thirty years is imposing pressure on performance analysts to provide reliable Quality of Service performance metrics. Analytical models are the preferred route to performance prediction in view of their precision and efficient execution, but tend to require overly restrictive assumptions, which we relax dramatically. We extend the M/M/1 Markovian queue specification to include geometrically distributed batches to model bursty processes, Markov modulation to model autocorrelation in traffic and switching between modes of operation, service breakdowns and repairs, negative customers to model losses and load balancing as well as multiple streams of all types of traffic described in these terms. We propose a solution mechanism, derived from the well-known spectral expansion and matrix geometric methods, that provides accurate state occupation probability (SOP) solutions for queues of this type at equilibrium. The development of our method first demonstrates and then overcomes weaknesses inherent in these basic methods when applied to our, more complex, queueing model. An automated implementation of the solution process allows for rapid deployment of what would otherwise be a cumbersome and error-prone process.

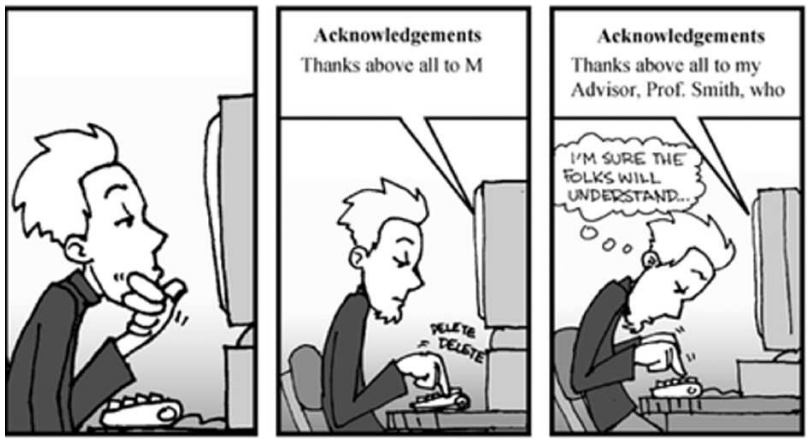
Quantiles on sojourn times, given by probability distribution functions, have recently become recognised as a critical metric for quality of service in computer networks as well as many other logistical systems. Using our equilibrium SOP results, we derive explicit expressions in the time domain for the sojourn (or response) time probability distribution, whereas previously, sojourn time distributions have not been obtained even for an MMPP/M/1 queue. We modify a previous result for their Laplace transform which we then show takes a rational form and can be inverted to give a mixture of exact distributions without the need for numerical inversion or approximation.

Single queue results are then deployed in a network setting with probabilistic routing, where an iterative algorithm for the approximate joint equilibrium state occupation probabilities of all queues is developed. Numerical results for both SOP and sojourn time distributions are provided that show good accuracy with respect to simulation in many networks and characteristics of networks where accuracy is poor are identified.

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible:

- My supervisor, Prof. Peter Harrison, who introduced me to the field and managed to keep me going throughout the process.
- My family for their love, encouragement and support.
- Dave Thornley, who gave me a starting point and worked with me until I was ready to branch out into my own material.
- Other members of the AESOP group: Ashok Argent-Katwala, Susanna Au-Yeung, Jeremy Bradley, Nicholas Dingle, Tony Field, William Knottenbelt, Ting Ting Lee, Uli Harder and Aleksandar Trifunovic. Discussions in and outside the pub allowed me to gain deeper insight into performance engineering-related and also many unrelated topics.
- The Engineering and Physical Sciences Research Council (EPSRC) for partially funding me.



"Piled Higher and Deeper" by *Jorge Cham* www.phdcomics.com

Contents

List of Figures

Chapter 1

Introduction

Sojourn times are an important metric in modern-day communication networks and other systems. Financial institutions for instance rely on incoming price feeds and outgoing orders to transfer funds within a certain minimal time interval. For this reason, it is of significant interest to accurately predict relative performances of competing solutions which promise to deliver dedicated, high-performance communication links.

Other uses are found in lower-end applications such as web servers and databases. Here it is also important to ensure that almost all user-requests are answered within certain, much less stringent, time bounds. While purchasing dedicated hardware or communication channels is often feasible, giving reliable bounds on timings, it can be desirable to utilise a public network like the Internet instead. This is especially the case if the increased uncertainty can be offset by the massive reduction in costs realised by avoiding a single-purpose line, if this uncertainty can be adequately described and bounded by rigorous analytic methods.

We present a type of queue which is amenable to modelling and analytic solutions both by itself and in a network context. Building on top of single-queue results by P.G. Harrison [Har02], sojourn times are made available without the need for error-prone calculations. Branching off from invaluable earlier work with D. Thornley [ThoZat+03], we develop an automated sojourn time generator for complex networks which is the first such contribution in the field.

1.1 Thesis Outline

Like Gaul, this thesis falls naturally into three parts: chapters 2 and 3 lead to steady-state occupation probabilities of queues. Chapter 4 finds queue response (or sojourn) time distributions and chapter 5 discusses both aspects within networks of queues.

Chapter 2 presents a versatile queueing paradigm, which is able to model burstiness by use of geometric batches as well as multi-mode arrivals and breakdowns and repairs of servers using Markov modulation, all of which are features frequently occurring in TCP/IP and other networks. Queues of this form are analysed in later chapters.

Chapter 3 shows that that, although the queueing model is rich in complexity, it lends itself to automated generation of steady-state state occupation probabilities (SOP) *i.e.* measures of the number of customers at a queueing node. We present currently used solution methods for simpler queue models to illustrate the procedures clearly and highlight, as well as overcome, difficulties that arise from the added complexity. The existence of readily available and exact results for SOPs is an imperative requirement for sojourn time calculations at equilibrium.

Chapter 4 describes the design and implementation of the automated sojourn time solver and is one of the main outputs of this research. Analytical models have struggled to provide precise estimates in the time domain and Laplace transforms of density functions have often been the best that can be determined. Although useful for calculating moments of distributions and often numerically invertible, quantiles can usually only be found reliably in the mid-range. In the crucial tail of a density function, numerical inversion quickly becomes unstable. By showing that the relevant Laplace transforms take rational, analytically invertible form we have obtained the full time-domain distribution functions.

Chapter 5 shifts the focus onto general networks of queues of the introduced type. Due to state space explosion it is generally prohibitive to solve for the joint SOP

of networks of queues. Although true product-form solutions exist occasionally, we can derive close approximations by using traffic matching between individual queues for network topologies which do not induce significant amounts of correlated traffic at different nodes. Queues inside the network are iteratively solved individually with emerging traffic passed between each other until the SOPs have settled down. For validation purposes, this chapter also introduces two types of simulators. The first accurately represents all actions within a given network and can give arbitrarily close approximations to the true joint SOP distributions for the constituent queues. The second supplies the theoretically most accurate inter-queue traffic approximation, as it represents emerging traffic by a general distribution function, whereas traffic matching is limited to representations using superimposed geometric streams only. As the calculations of sojourn times in networks depend on the results of the traffic-matching stage, this step is important to identify network topologies which do not lend themselves to being analysed in this decompositional manner.

Chapter 6 summarises the contributions of the thesis, presents conclusions and considers applications as well as avenues for future work.

Appendix A details the localisation procedure that is applied to the balance equations of chapter 2 to make them amenable to analysis in subsequent chapters.

Appendix B gives a short overview of the syntax of PEPA, a stochastic process algebra, that is use to describe networks in chapter 5.

1.2 Notation

To achieve clarity of expression in the multitude of formulae throughout the chapters, certain conventions will be followed that allow the deduction of the type of an object by the style it is written in.

Vectors are written in bold, usually lowercase, letters with arrows pointing to the right on top. Examples would be $\overrightarrow{\mathbf{v}}$ and $\overrightarrow{\lambda}$. A few useful special cases are used throughout:

$\overrightarrow{\mathbf{0}}$ is a zero-vector and $\overrightarrow{\mathbf{e}}_n$ is the n^{th} unit vector, *i.e.* the vector that has component 1 at position n and a zero component otherwise. Finally, $\overrightarrow{\mathbf{e}}$ is a vector of all-ones, *i.e.* $\overrightarrow{\mathbf{e}} = \sum_n \overrightarrow{\mathbf{e}}_n$. All vectors are assumed to be column vectors. When row vectors are needed, they are transposed in the standard way: $\overrightarrow{\mathbf{v}}^T$ is the transpose of $\overrightarrow{\mathbf{v}}$.

Matrices are capital letters that are bold and underlined, such as $\underline{\mathbf{A}}$ and $\underline{\mathbf{\Theta}}$. An exception to this rule is $\underline{\mathbf{0}}$, which denotes the zero-matrix. As is customary elsewhere, $\underline{\mathbf{I}}$ is the identity matrix. The vector operator $\underline{\mathbf{diag}}(\overrightarrow{\mathbf{x}})$ produces a diagonal matrix with the vector elements on its diagonal.

Unless explicitly stated, all symbols that do not fit the previous criteria are scalar quantities. As part of some of the approximation processes in later chapters, we will frequently encounter situations where mean quantities are needed. The specifics of what quantity the mean is calculated over will be stated when needed, but a common feature of all mean quantities is the use of an overline, $\overline{s}$, to hint at its property. Such means can also be applied to non-scalar quantities, so that $\overline{\overrightarrow{\mathbf{s}}}$ would be a component-wise average vector.

Similarly, a triangular hat above a symbol, $\underline{\hat{\mathbf{F}}}$, indicates that we are dealing with a quantity that has been found numerically, using *e.g.* iterative methods. These quantities contain numerical errors and are used during accuracy analysis.

Whenever key terms are mentioned for the first time, they are usually written in *italic* and added to the glossary at the end of the thesis.

1.3 Statement of Originality and Publications

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

The queue model presented in chapter 2 as well as the associated automated solution mechanism outlined in Appendix A were motivated by an EPSRC research grant (MEGAN), and developed together with David Thornley. These results have been published at the first international working conference on performance modelling and

evaluation of heterogeneous networks (HET-NETs 2003) [ThoZat+03]. An extended version of this paper is now being considered to appear in the Annals of Operations Research.

Sojourn time distributions given in chapter 4 form the basis for a part of a paper at the 1st International Conference on Quantitative Evaluation of SysTems (QEST) September 2004, University of Twente [HarZat04]. The other part of said paper concerns itself with a simplification of a previous result in [Har02] and was contributed by Peter G Harrison. It does not appear in this thesis.

Network results of chapter 5 have been published in parts at the 19th UK Performance Engineering Workshop (UKPEW' 2003), University of Warwick [ThoZat03], as well as the seventeenth International Symposium on Computer and Information Sciences October 2002, University of Central Florida [HarTho+02].

Chapter 2

Queue type

2.1 Introduction

We will be modelling the service centres within a network as queues. Our aim is to provide a powerfully descriptive, yet still tractable model of actual service centres. Simpler types of queueing networks have already been extensively studied (see *e.g.* [Mit87, HarPat92]) and analytical expressions for many performance measures are already known. For our enhanced model we wish to in particular represent burstiness, failures (and subsequent repairs) of servers, qualitative changes in traffic external to the server as well as provisions for inter-server interactions depending on their load. The key to the model remaining soluble using standard methods lies in only allowing Markovian transitions within the resulting complex structure. The memoryless property then ensures that the solution process of the resulting finite (or infinite) Markov chain can be handled by existing methods [MitCha95, BinLat+02].

The detailed description of the transition structure for the queue-model is given by way of *Kolmogorov balance equations* (*c.f. e.g.* [HarPat92]). For each state the queue can occupy, these balance equations equate the probability flux into the state with that which is going out of it. At steady-state, *i.e.* when the system has settled down, these two measures must be equal, so that this forms a necessary condition for our solution, which is also sufficient by the Steady-State Theorem for Markov chains [HarPat92].

To aid the understanding of the complexities that are brought by the queue features, we shall begin with the Kolmogorov balance equations for the simplest type of queue, written $M/M/1/L$ in Kendall notation. This notation allows the description of important queue features in a succinct form.

The first part of the name qualitatively describes the arrival process at a queue, which in our case is M or Markovian (Poisson with exponential inter-arrival times). Other possibilities would be D for deterministic or G for general. We will using Markovian processes exclusively, as these are the simplest to analyse, yet still reflect actual behaviour. The second M represents the service process that the queue provides. Again, the choices include Markovian, deterministic and general. In third position, we have the number c of servers within the queue. When this number is greater than 1, it is assumed that the servers operate in an indistinguishable manner, *i.e.* the servers are homogeneous. Finally, the L is a scalar that describes the maximum possible queue length, in case a queue has a limited size buffer. When the queue under consideration is infinite, *i.e.* $L = \infty$, this part is usually omitted and we use the term $M/M/1$.

For each new concept we include above and beyond the $M/M/1/L$ queue we successively augment the representation until the full balance equations for all the modelled concepts are derived.

In all the following, the balance equations are written in the form $r_j = 0$ for $0 \leq j \leq L$. The left-hand-side (LHS) of the j^{th} Kolmogorov balance equation r_j for the $M/M/1/L$ queue is given by:

$$r_j^{\text{M/M/1/L}} = \pi_{j-1}\lambda f_{j>0} - \pi_j\left[\lambda f_{j<L} + \mu f_{j>0}\right] + \pi_{j+1}\mu f_{j<L} \tag{2.1}$$

This expression gives the difference between the incoming and outgoing probability fluxes at level j, where π_j is the equilibrium probability of occupying state j, and f is the indicator function defined by $f_{\text{true}} = 1$ and $f_{\text{false}} = 0$. The terms λ and μ are the arrival and service rates respectively.

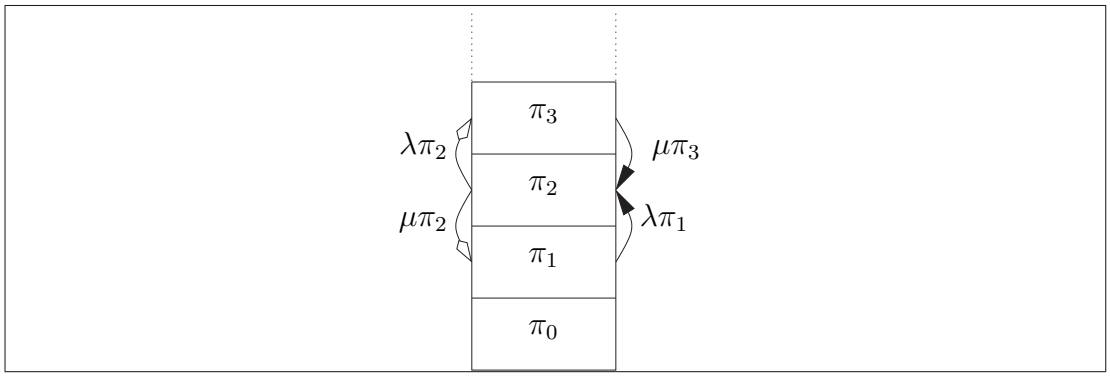


Figure 2.1: Sample transitions for queue level $j = 2$ within the $M/M/1$ queue.

2.2 Modulation

We allow the current state of an independent continuous time, discrete state Markov process to select the parameters for the interval distributions of the arrivals and service completions. Such MMPPs (Markov Modulated Poisson Processes) [FisMei92, CoxIsh 80] have been applied to queueing systems since [Neu71]. Queues with purely Poisson arrivals and departures are known not to accurately represent network traffic [LelTaq+94, PaxFlo95], yet MMPPs have recently been used with some success in representing real-world Ethernet traffic [GeHar+03, Mei87, DenMar93, AsmNer+96, Ryd96]. The transitions between modulation states which we also call *phases*, is fully defined by its instantaneous transition rate (generator) matrix $\underline{\mathbf{Q}}$:

$$\underline{\mathbf{Q}} = \begin{pmatrix} -\sum_{i\neq 1} q_{1i} & q_{12} & \cdots & q_{1N} \\ q_{21} & -\sum_{i\neq 2} q_{2i} & & q_{2N} \\ \vdots & & \ddots & \vdots \\ q_{N1} & \cdots\cdots\cdots\cdots & & -\sum_{i\neq N} q_{Ni} \end{pmatrix}$$

Here, the entries $q_{ij} \geq 0$ give the rate at which the modulation process switches from state i to state j. Not all values for q_{ij} lead to a valid solution at steady-state. The matrix $\underline{\mathbf{Q}}$ in (2.2) gives a well-defined modulation process, but it does not have a steady-state solution because its state space is not connected. As a result the long-term behaviour depends on the initial occupied state, which is incompatible with steady-state analysis.

$$\underline{\mathbf{Q}} = \begin{pmatrix} -3 & 3 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -2 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{2.2}$$

For the remainder of this thesis it will be assumed that all matrices $\underline{\mathbf{Q}}$ do not exhibit this behaviour and yield a unique steady-state probability distribution.

Together, the queue length and modulation states of the resulting queue form a 2D lattice strip. The dimension describing the state of the modulator is finite and the other dimension, the queue-length, may be finite or infinite. The equilibrium probability of being in any particular column of the lattice is given by the steady-state phase probability of the modulation process $\overrightarrow{\pi} = (\pi_1, \ldots, \pi_N)$ defined by the two equations $\overrightarrow{\pi}.\underline{\mathbf{Q}} = \overrightarrow{\mathbf{0}}$, and $\|\overrightarrow{\pi}\|_1 = 1$.[1]

To allow for modulation in our model we treat queue levels as single entities by representing the state occupation probability (SOP) at level j as the N-component row vector $\overrightarrow{\mathbf{v}}_j$. This vector consists of the SOPs of the N modulation phases at that level. In addition, the representation of arrivals and services is augmented to give an analogous vector equation.

A two state modulated system with an arrival stream of positive customers with rates λ_1 and λ_2 and a service stream with rates μ_1 and μ_2 is represented using the following matrices

$$\underline{\mathbf{\Lambda}} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \underline{\mathbf{M}} = \begin{pmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{pmatrix}$$

This notation allows us to multiply SOP vectors on a lattice row by a matrix to give standard-looking rate terms, with the substitution of the identity matrix $\underline{\mathbf{I}}$ for 1. For the j^{th} balance equation this gives

$$\overrightarrow{\mathbf{r}}_j^{\text{MM/MM/1/L}} = \overrightarrow{\mathbf{v}}_{j-1}\underline{\mathbf{\Lambda}} f_{j>0} + \overrightarrow{\mathbf{v}}_j \left[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} f_{j<L} - \underline{\mathbf{M}} f_{j>0}\right] + \overrightarrow{\mathbf{v}}_{j+1}\underline{\mathbf{M}} f_{j<L}$$

[1] We express the sum of elements in a vector $\overrightarrow{\mathbf{x}}$ as its L_1 norm $\left\|\overrightarrow{\mathbf{x}}\right\|_1 = \sum_i |x_i|$

Compare this expression with that for the unmodulated queue shown in (2.1) to see the substitution of vectors and matrices as appropriate for SOPs and rates respectively. The inclusion of $\underline{\mathbf{Q}}$ provides the modulation transitions.

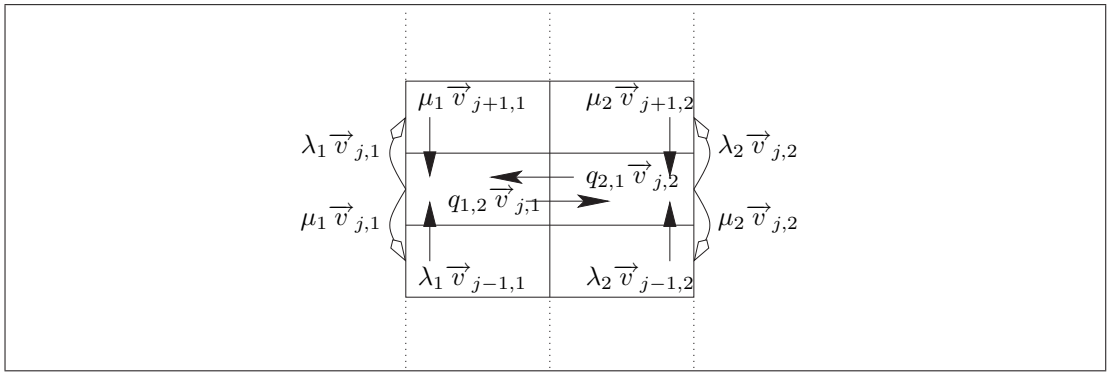


Figure 2.2: Sample transitions for a Markov modulated queue with $N = 2$ modulation states.

2.3 Geometric batches and the GE distribution

The compound Poisson process (CPP) describes point arrivals of customers at exponentially distributed intervals. Every such point arrival yields a batch arrival of customers, which – in our model – are geometrically distributed; this can model burstiness (*e.g.* [BhaHar97]). The generalised exponential (GE) distribution of processing times describes a similar batched behaviour for service completions. The probability distribution function F_{GE} of the inter-arrival time random variable T of a CPP is a "generalised exponential" distribution [KouAwa03].

$$F_{GE}(t) = P(T \leq t) = 1 - (1 - \theta)e^{-\lambda'(1-\theta)t}$$

We desire to express the inter-arrival time exclusively through a parameter λ, whereas the batch distributions should only depend on the parameter θ. Therefore, we use the particular value $\lambda = \lambda'(1 - \theta)$: $F_{GE}(t) = P(T \leq t) = 1 - (1 - \theta)e^{-\lambda t}$.

The $(1 - \theta)$ term gives an impulse at the origin ($F_{GE}(0) = \theta$) giving a non-zero probability to a zero inter-arrival time. This allows a sequence of one or more zero

inter-arrival times, and hence non-unit (geometric) batches. A deterministic, unit batch size is given by setting the geometric distribution parameter to zero.

The queue can accommodate geometrically batched occurrences of both arrivals and processing completions. These batched streams cause transitions within the queue whose horizontal component is zero and with vertical component (increasing or decreasing queue length) given by a, possibly truncated, geometric distribution.

An arrival stream is truncated when it operates in a finite queue. When the arrival of a batch of customers would overflow the queue, *i.e.* the resulting number of customers within the queue would exceed the maximum queue length, we discard the excess customers and make the transition to the top of the queue. Hence, the arrival batch size s is distributed as follows

$$P(s = S|j = J) = \begin{cases} (1-\theta)\theta^{S-1} & \text{if } S + J < L \\ \theta^{S-1} & \text{if } S + J = L \\ 0 & \text{otherwise} \end{cases}$$

It follows that arrival streams to infinite queues, where $L = \infty$, are never truncated.

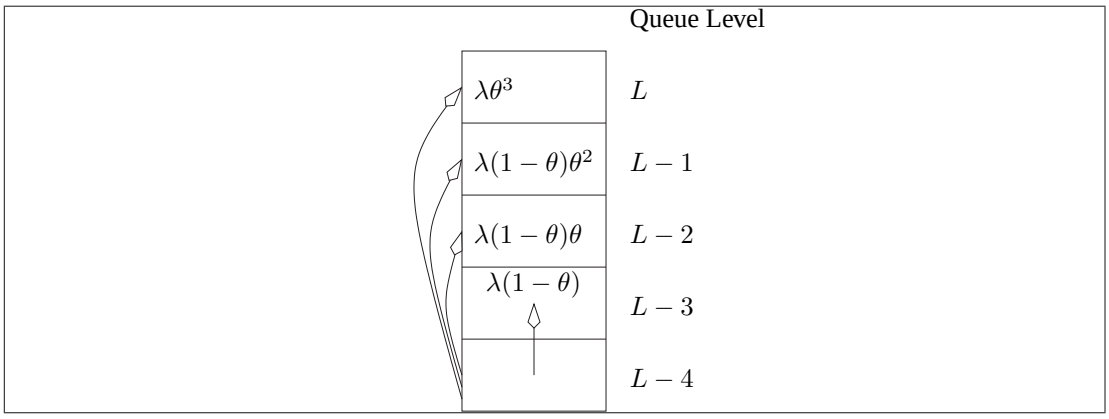


Figure 2.3: Arrival process truncation from level $L - 4$ for a finite queue.

Batch distributions of service completions also need to be truncated. Unlike the case for arrival streams this truncation is not dependent on some queue parameter, but occurs unconditionally, because the size of a service completion batch can never exceed the current queue length. As a result the distribution for the truncated batch sizes of

service completions is given by

$$P(s = S|j = J) = \begin{cases} (1-\phi)\phi^{S-1} & \text{if } S < J \\ \phi^{S-1} & \text{if } S = J \\ 0 & \text{otherwise} \end{cases}$$

As with the rates $\underline{\mathbf{\Lambda}}$ and $\underline{\mathbf{M}}$ we write the batch parameters in matrix form, where $\underline{\mathbf{\Theta}}$ is the diagonal matrix of geometric arrival batch size parameters (θ) and $\underline{\mathbf{\Phi}}$ similarly for service batch sizes (ϕ). For a two-state system the matrices would be

$$\underline{\mathbf{\Theta}} = \begin{pmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{pmatrix}, \underline{\mathbf{\Phi}} = \begin{pmatrix} \phi_1 & 0 \\ 0 & \phi_2 \end{pmatrix}$$

Now we wish to obtain an expression giving the LHS balance equation when geometric arrival and service batches are included. To achieve a unified expression for this term we again use the indicator or "flag" function defined by $f_{true} = 1$ and $f_{false} = 0$. The terms $f_{j<L}$ and $f_{j>0}$ then identify those levels where truncation *does not* happen, in this case we speak of *unbounded* batch flow. Conversely, when we are at a level at which truncation does happen for a given stream the value of these same flags is zero. The values these flags take are then used to raise matrices to a power, so that the $(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})$ and $(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})$ terms only appear where there is *unbounded* batch flow. Using the convention that $\underline{\mathbf{A}}^0 = \underline{\mathbf{I}}$ and $\underline{\mathbf{A}}^1 = \underline{\mathbf{A}}$ for all matrices $\underline{\mathbf{A}}$, the LHS balance equation now become

$$\begin{aligned} \overrightarrow{\mathbf{r}}_j^{\text{MM CPP/GE/1/L}} = & \sum_{i=0}^{j-1} \overrightarrow{\mathbf{v}}_i \underline{\mathbf{\Lambda}} (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})^{f_{j<L}} \underline{\mathbf{\Theta}}^{j-i-1} \\ & + \overrightarrow{\mathbf{v}}_j \left[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} f_{j<L} - \underline{\mathbf{M}} f_{j>0} \right] \\ & + \sum_{i=j+1}^{L} \overrightarrow{\mathbf{v}}_i \underline{\mathbf{M}} (\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})^{f_{j>0}} \underline{\mathbf{\Phi}}^{i-j-1} \end{aligned}$$

2.4 Multiple Processors

So far our model has assumed the presence of exactly one processor ($c = 1$), with its processing rates given by the matrix $\underline{\mathbf{M}}$ and batches described by the matrix $\underline{\mathbf{\Phi}}$. To

introduce multiple homogeneous processors, the processing rate matrix $\underline{\mathbf{M}}$ is replaced by $\underline{\mathbf{C}}_j = \min(j, c)\underline{\mathbf{M}}$ at queue length j, with the batch matrix $\underline{\mathbf{\Phi}}$ remaining unchanged. If all processors are busy, then service batches can clear jobs down to level $c-1$ inclusive. In other words there is *unbounded* batch flow to levels c and above, and *bounded* (truncated) batch flow to $c-1$. This generalisation of the truncation level compared to the $c = 1$ case treated before is carried out by changing the arguments of the indicator functions. Should any of the processors be idle, the processing batch size is exactly 1 as there are no jobs in the waiting room and a processor can only clear its own job in service. The LHS is thus:

$$\begin{aligned}\overrightarrow{\mathbf{r}}_j^{\text{MM CPP/GE/c/L}} = &\sum_{i=0}^{j-1} \overrightarrow{\mathbf{v}}_i \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})^{f_{j<L}} \underline{\mathbf{\Theta}}^{j-i-1} \\ &+ \overrightarrow{\mathbf{v}}_j \left[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} f_{j<L} - \underline{\mathbf{C}}_j f_{j>0}\right] \\ &+ \sum_{i=j+1}^{L} \overrightarrow{\mathbf{v}}_i \underline{\mathbf{C}}_i (\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})^{f_{j>c-1}} \underline{\mathbf{\Phi}}^{i-j-1} f_{\substack{(i=j+1) \\ \vee (j \geq c-1)}}\end{aligned}$$

The indicator function $f_{j>c-1}$ bounds downward flow at level $c-1$, whereas the term $f_{(i=j+1)\vee(j\geq c-1)}$ selects valid flows, which are batches from any queue length to just below c, or single jobs from a single processor.

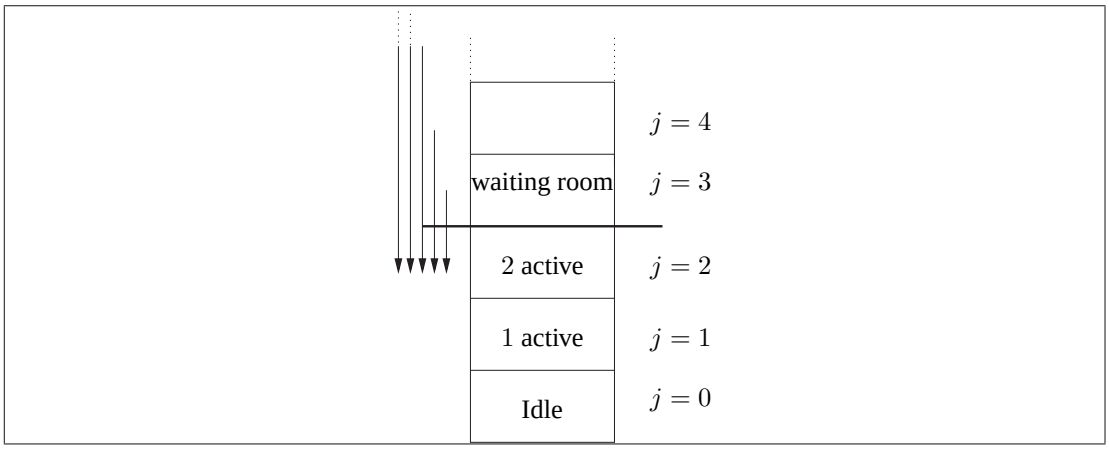


Figure 2.4: Service batch truncation at level $c-1$ for multi-servers. Note that level $j = 1$ is not reachable through a batch service completion.

2.5 Breakdowns and Repairs

We treat breakdowns and repairs (as defined by Mitrani and Chakka [MitCha95]) by allowing the number of processors to vary from 0 to c across the modulation phases. Thus the number of phases is $N = c + 1$. This is introduced into the left-hand side expression by replacing references to c with a vector, $\overrightarrow{\mathbf{c}}$, of the numbers of operational processors in each modulation state, so that

$$\overrightarrow{\mathbf{c}} = (0, 1, \ldots, N) = (c_1, c_2, \ldots, c_{N+1})$$

and

$$\underline{\mathbf{C}}_j = \underline{\mathbf{diag}}(\min(j, c_i))\underline{\mathbf{M}}$$

Our LHS expression becomes

$$\begin{aligned}\overrightarrow{\mathbf{r}}_j^{\text{MM CPP/GE/Mc/L}} = &\sum_{i=0}^{j-1} \overrightarrow{\mathbf{v}}_i \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})^{f_{j<L}} \underline{\mathbf{\Theta}}^{j-i-1} \\ &+ \overrightarrow{\mathbf{v}}_j \left[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} f_{j<L} - \underline{\mathbf{C}}_j f_{j>0}\right] \\ &+ \sum_{i=j+1}^{L} \overrightarrow{\mathbf{v}}_i \underline{\mathbf{C}}_i (\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})^{\underline{\mathbf{F}}_{j>c_m-1}} \underline{\mathbf{\Phi}}^{i-j-1} f_{\substack{(i=j+1) \\ \vee (j \geq c_m - 1)}}\end{aligned}$$

where $\underline{\mathbf{F}}_{\overrightarrow{\mathbf{p}}}$ is a diagonal matrix of values whose i^{th} diagonal element is f_{p_i}. We define the result of raising a square matrix $\underline{\mathbf{A}}$ to the power $\underline{\mathbf{B}}$ with the same dimensions to be a corresponding matrix of elements $a_{i,j}^{b_{i,j}}$. In fact all matrices operated upon in this manner are diagonal. We combine breakdowns and repairs with modulated arrivals by the standard technique of taking the Kronecker product of the independent modulation matrices.

2.6 Negative customers

Queues with negative customers, so called *G-queues* [Gel91], experience additional downward flux which can be used to model network phenomena such as losses, killing

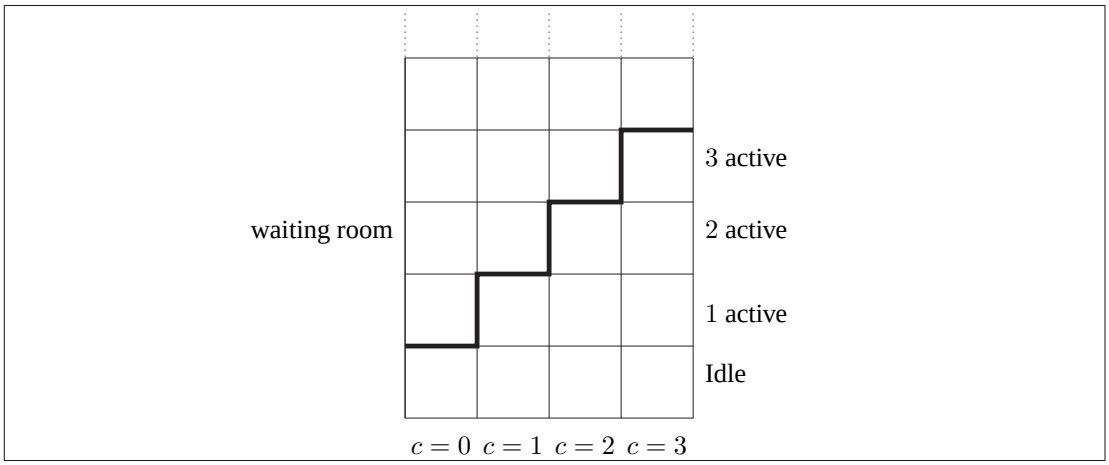


Figure 2.5: A $c = 3$ server with breakdowns and repairs. States above the thick line have customers in the waiting room.

signals in speculative parallelism and load balancing. In addition to the diagonal negative arrival rate and batch size matrices $\underline{\mathbf{K}}$ and $\underline{\mathbf{R}}$, a killing mode has to be chosen. We allow for three modes: t^v or *tail vulnerable* removes a job from the tail of the queue even if it is in service, t^s or *tail safe* removes a job from the tail of the queue but not when in service, and h^p or *head per* removes a customer from the head of the queue (in service) at an independent but equal rate per processor, leading to a lower total loss rate when some processors are inactive.

Killing mode t^v is the simplest in a sense as it can kill any job in or out of service and the batches are bounded only at level zero. Mode t^s is bounded at level c, as it cannot kill any job in service. Mode h^p causes flux identical to processing completions. Contributing these killing modes, we obtain

$$\begin{aligned}
\overrightarrow{\mathbf{r}}_j = & \sum_{i=0}^{j-1} \overrightarrow{\mathbf{v}}_i \big[\quad \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})^{f_{j<L}} \underline{\mathbf{\Theta}}^{j-i-1} \big] \\
& + \overrightarrow{\mathbf{v}}_j \big[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}}\, f_{j<L} - \underline{\mathbf{K}}\, f_{((j>\kappa^b)\vee \mathrm{h}^p)} \underline{\beta}_j - \underline{\mathbf{C}}_j \big] \\
& + \sum_{i=j+1}^{L} \overrightarrow{\mathbf{v}}_i \big[\quad \underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})^{f_{j>\kappa^b}} \underline{\mathbf{R}}^{i-j-1} \quad f_{\substack{(j\geq\kappa^b) \\ \vee(\mathrm{h}^p \wedge i=j+1)}} \quad \beta_i \\
& \qquad + \underline{\mathbf{C}}_i (\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})^{f_{j>c-1}} \underline{\mathbf{\Phi}}^{i-j-1} \, f_{\substack{(i=j+1) \\ \vee(j\geq c-1)}} \big]
\end{aligned}$$

where κ^b is the lowest level reachable by killing, *i.e.* ${\kappa^b}_m = c_m f_{\mathrm{t}^s} + (c_m - 1) f_{\mathrm{h}^p}$, at which batch killing is truncated. We define the m^{th} element of the diagonal $N \times N$

killing factor matrix at level (queue length) j, $\underline{\beta}_{\mathrm{j}\,m,m} = \min(j, c_m)/\max_m(c_m)$ for h^p killing and $c_m/\max_m(c_m)$ otherwise.

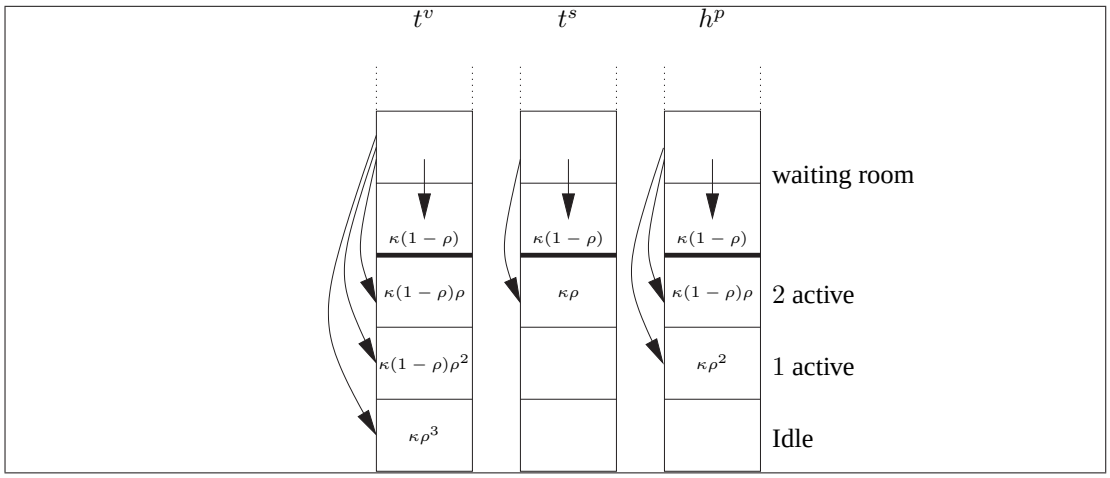


Figure 2.6: The three different killing modes illustrated on a non-modulated, $c = 2$ multi-server queue. The levels at which the negative customer batches are truncated at 0, c and $c-1$ respectively.

2.7 Multiple streams

In order to add multiple streams we augment the relevant term to be a sum of streams. For multiple arrivals, this would take the form of

$$\begin{aligned}\overrightarrow{\mathbf{r}}_j^{\text{MM CPP}_k\ldots} &= \textstyle\sum_{i=0}^{j-1} \overrightarrow{\mathbf{v}}_i \sum_{k=1}^{n^{arr}} \underline{\mathbf{\Lambda}}_k (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}_k)^{f_{j<L}} \underline{\mathbf{\Theta}}_k^{j-i-1} \\ &+ \overrightarrow{\mathbf{v}}_j \left[\underline{\mathbf{Q}} - \textstyle\sum_{k=1}^{n^{arr}} \underline{\mathbf{\Lambda}}_k \, f_{j<L} - \underline{\mathbf{K}} \, f_{((j>\kappa^b)\vee \mathrm{h}^p)} \underline{\beta}_j - \underline{\mathbf{C}}_j\right] \\ &+ \ldots \text{(same terms as before)}\end{aligned}$$

where n^{arr} is the number of arrival streams. In an analogous way this introduction of sums of streams is also be realised for n^{serv} processing streams and n^{kill} negative customer streams to create the general balance equation.

$$\begin{aligned}\overrightarrow{\mathbf{r}}_j^{general} = & \sum_{i=0}^{j-1} \overrightarrow{\mathbf{v}}_i \big[\quad \textstyle\sum_{k=1}^{n^{arr}} \underline{\mathbf{\Lambda}}_k (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}_k)^{f_{j<L}} \underline{\mathbf{\Theta}}_k{}^{j-i-1} \big] \\ & + \overrightarrow{\mathbf{v}}_j \big[\underline{\mathbf{Q}} - \textstyle\sum_{k=1}^{n^{arr}} \underline{\mathbf{\Lambda}}_k \, f_{j<L} - \sum_{k=1}^{n^{kill}} \underline{\mathbf{K}} \, f_{((j>\kappa^b)\vee \mathrm{h}^p)} \beta_j - \sum_{k=1}^{n^{serv}} \underline{\mathbf{C}}_{k,j} \big] \\ & + \sum_{i=j+1}^{L} \overrightarrow{\mathbf{v}}_i \big[\quad \textstyle\sum_{k=1}^{n^{kill}} \underline{\mathbf{K}}_k (\underline{\mathbf{I}} - \underline{\mathbf{R}}_k)^{f_{j>\kappa^b}} \underline{\mathbf{R}}_k{}^{i-j-1} \quad f_{\substack{(j\geq\kappa^b) \\ \vee(\mathrm{h}^p \wedge i=j+1)}} \beta_i \\ & \qquad + \textstyle\sum_{k=1}^{n^{serv}} \underline{\mathbf{C}}_{k,i} (\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}_k)^{f_{j>c-1}} \underline{\mathbf{\Phi}}_k{}^{i-j-1} \, f_{\substack{(i=j+1) \\ \vee(j\geq c-1)}} \big] \end{aligned}$$

2.8 Normalisation

We now have a set of balance equations $\overrightarrow{\mathbf{r}}_j = 0$ with unknowns $\overrightarrow{\mathbf{v}}_j$, where $0 \leq j \leq L$. At steady-state this set forms a necessary condition on the probability distribution given by the $\overrightarrow{\mathbf{v}}$'s, if it exists. This linear homogeneous system, with the same number of equations as unknowns, does not have a unique solution, but we replace (any) one of them by the additional equation given by the normalisation condition *i.e.* an equation that ensures that the sum of all occupation probabilities $\overrightarrow{\mathbf{v}}$ is 1. The given balance equations are vector equations, so in practice we replace the i^{th} component for some $1 \leq i \leq N$.

$$\sum_{j=0}^{L} \left\| \overrightarrow{\mathbf{v}}_j \right\|_1 = 1$$

If the chosen parameters for rates and batches permit a solution of these equations, this solution is unique by the Steady-State Theorem for Markov chains [HarPat92] and the queue is said to be solvable.

Chapter 3

Steady-state solving

3.1 Introduction

The ensemble of balance equations combined with the normalisation constraint can now be used to find the (unique) solution for the steady-state probabilities $\overrightarrow{\mathbf{v}}_j$. This is easily done directly, using for example Gaussian elimination, if the represented Markov chain has a small state space. The size of this state space is equal to the product of the number of modulation states, N, and the number of queue levels (counting from zero), $L + 1$. In practice, N will generally be small, and certainly finite, whereas for the queue capacity L, large values or even infinity are common. In these cases it is not possible to use direct methods as they are computationally too expensive or not applicable.

There are two existing methods that solve comparable, but simpler, problems with large state spaces. The Matrix Geometric (MG) [BinLat+02] method and the spectral expansion (SE) [Cha95] method both seek to represent a region of queue length probabilities using geometric series. Both require that the Kolmogorov balance equations within this region form a low-order linear homogeneous matrix recurrence relation in the equilibrium occupation probability vectors at adjacent queue lengths. The solution to such a system is then a sum of geometric series, which may be generated explicitly and individually as in spectral expansion, or grouped and assembled into matrices as

in matrix geometric methods.

However, both methods are not immediately applicable to our set of Kolmogorov balance equations, because the balance equations (in section 2.7) are distinct for any two levels and they do not form the required linear homogeneous matrix recurrence relation. In addition it is apparent that due to the presence of unbounded geometric batches each individual balance equation involves, in general, probabilities $\overrightarrow{\mathbf{v}}_j$ from throughout the range of queue lengths j.

To enable the use of both the MG and SE methods on our queue model, we have developed a methodology that applies linear transformations to the balance equations in order to transform them into an equivalent set of *localised* balance equations. The key property of localised balance equations is that they only involve the probability vectors $\overrightarrow{\mathbf{v}}_j$ for a small, finite range of j. Localisation is achieved by adding appropriate scalar multiples of adjacent balance equations. This process differs from Gaussian elimination in that it takes advantage of the special structure of the balance equations, allowing the cancellation of all but one of the terms from a batched stream using only one symbolic operation. The localisation procedure is described in detail in Appendix A.

Key to using these transformed equations for the solution process is the large, possibly infinite[1], *repeating region* within the set of queue lengths for which the localised balance equations at level j are of identical form: they only involve dependencies on the probabilities $\overrightarrow{\mathbf{v}}_j$ (*i.e.* parameterised by j) at $n^{serv} + n^{kill}$ levels above and n^{arr} levels below j. Thus, our localisation procedure yields a homogeneous linear matrix recurrence of order $n^{arr} + n^{serv} + n^{kill}$, which we will write as

$$\sum_{k=0}^{n^{arr}+n^{serv}+n^{kill}} \overrightarrow{\mathbf{v}}_{j+k-n^{arr}} Q_k = \overrightarrow{\mathbf{0}} \quad \text{(for } j \text{ within the repeating region)} \tag{3.1}$$

3.2 Example Matrix recurrence

We provide a set of example balance equation, that would apply in a $c = 1$, $L = 6$ queue with one geometrically batched arrival stream ($n^{arr} = 1$), one geometrically

[1] only in case of infinite queues

batched service stream ($n^{serv} = 1$) and no negative customers ($n^{kill} = 0$). When localised using the method in Appendix A or [ThoZat+03], the balance equations become as follows in the order of levels to which they pertain, from 0 to 6:

$$\begin{aligned}
\overrightarrow{\mathbf{v}}_0(\underline{\mathbf{Q}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}) + \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Theta\Phi}} - \underline{\mathbf{I}}) + \overrightarrow{\mathbf{v}}_1(\underline{\mathbf{M}} + \underline{\mathbf{\Lambda\Phi}} - \underline{\mathbf{Q}}\underline{\mathbf{\Phi}}) &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\mathbf{v}}_0(\underline{\mathbf{\Lambda}}(\underline{\mathbf{\Theta}} - \underline{\mathbf{I}})(\underline{\mathbf{\Theta\Phi}} - \underline{\mathbf{I}})) + \overrightarrow{\mathbf{v}}_1(\underline{\mathbf{Q}} - \underline{\mathbf{M}} + \underline{\mathbf{\Lambda}}((\underline{\mathbf{\Theta}} - \underline{\mathbf{I}})\underline{\mathbf{\Phi}} - \underline{\mathbf{I}}))+ & \\
\overrightarrow{\mathbf{v}}_2(\underline{\mathbf{M}} - \underline{\mathbf{Q}}\underline{\mathbf{\Phi}} + \underline{\mathbf{\Lambda\Phi}}) &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\mathbf{v}}_1(\underline{\mathbf{M\Theta}} - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}} + \underline{\mathbf{\Lambda}}) + \overrightarrow{\mathbf{v}}_2(\underline{\mathbf{Q}}(\underline{\mathbf{\Theta\Phi}} + \underline{\mathbf{I}}) - \underline{\mathbf{M}}(\underline{\mathbf{\Theta}} + \underline{\mathbf{I}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{I}}))+ & \\
\overrightarrow{\mathbf{v}}_3(\underline{\mathbf{M}} - \underline{\mathbf{Q}}\underline{\mathbf{\Phi}} + \underline{\mathbf{\Lambda\Phi}}) &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\mathbf{v}}_2(\underline{\mathbf{M\Theta}} - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}} + \underline{\mathbf{\Lambda}}) + \overrightarrow{\mathbf{v}}_3(\underline{\mathbf{Q}}(\underline{\mathbf{\Theta\Phi}} + \underline{\mathbf{I}}) - \underline{\mathbf{M}}(\underline{\mathbf{\Theta}} + \underline{\mathbf{I}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{I}}))+ & \\
\overrightarrow{\mathbf{v}}_4(\underline{\mathbf{M}} - \underline{\mathbf{Q}}\underline{\mathbf{\Phi}} + \underline{\mathbf{\Lambda\Phi}}) &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\mathbf{v}}_3(\underline{\mathbf{M\Theta}} - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}} + \underline{\mathbf{\Lambda}}) + \overrightarrow{\mathbf{v}}_4(\underline{\mathbf{Q}}(\underline{\mathbf{\Theta\Phi}} + \underline{\mathbf{I}}) - \underline{\mathbf{M}}(\underline{\mathbf{\Theta}} + \underline{\mathbf{I}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{I}}))+ & \\
\overrightarrow{\mathbf{v}}_5(\underline{\mathbf{M}} - \underline{\mathbf{Q}}\underline{\mathbf{\Phi}} + \underline{\mathbf{\Lambda\Phi}}) &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\mathbf{v}}_4(\underline{\mathbf{M\Theta}} - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}} + \underline{\mathbf{\Lambda}}) + \overrightarrow{\mathbf{v}}_5(\underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{\Phi}} - \underline{\mathbf{I}})) - \underline{\mathbf{M}} - \underline{\mathbf{\Lambda\Phi}})+ & \\
\overrightarrow{\mathbf{v}}_6(\underline{\mathbf{Q}}(\underline{\mathbf{\Theta}} - \underline{\mathbf{I}})(\underline{\mathbf{\Phi}} - \underline{\mathbf{I}})) &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\mathbf{v}}_5(\underline{\mathbf{M\Theta}} - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}} + \underline{\mathbf{\Lambda}}) + \overrightarrow{\mathbf{v}}_6(\underline{\mathbf{Q}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}) + \underline{\mathbf{M}}(\underline{\mathbf{\Theta\Phi}} - \underline{\mathbf{I}})) &= \overrightarrow{\mathbf{0}}
\end{aligned}$$

For this example, it is apparent that the localisation procedure provides balance equations of identical form for levels 2, 3 and 4: This is our repeating region. The matrix recurrence (3.1) that applies within it has the following $\underline{\mathbf{Q}}_i$-terms:

$$\begin{aligned}
\underline{\mathbf{Q}}_0 &= \underline{\mathbf{M\Theta}} - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}} + \underline{\mathbf{\Lambda}} \\
\underline{\mathbf{Q}}_1 &= \underline{\mathbf{Q}}(\underline{\mathbf{\Theta\Phi}} + \underline{\mathbf{I}}) - \underline{\mathbf{M}}(\underline{\mathbf{\Theta}} + \underline{\mathbf{I}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{I}}) \\
\underline{\mathbf{Q}}_2 &= \underline{\mathbf{M}} - \underline{\mathbf{Q}}\underline{\mathbf{\Phi}} + \underline{\mathbf{\Lambda\Phi}}
\end{aligned} \tag{3.2}$$

The size of the repeating region within which this recurrence is valid depends on the total number of streams. For a queue of length L with c processors and no negative customers, the repeating region stretches from $j = c + n^{arr}$ to $j = L - 1 - n^{serv}$. We call the lower and upper bounds of the repeating region ϵ^b and ϵ^t, so that here, $\epsilon^b = c + n^{arr}$ and $\epsilon^t = L - 1 - n^{serv}$. When all the negative customer streams are of type tail-vulnerable (tv) or head-per (hp), they have the same impact on the size of the repeating region as if they were service streams and the repeating region spans

$$\epsilon^b = c + n^{arr} \leq j \leq L - 1 - n^{serv} - n^{kill} = \epsilon^t$$

When the queue has one or more tail-safe (ts) killing streams, the repeating region is one level smaller due to the fact that its batches are truncated to level c (*c.f.* section 2.6), which is one level higher than the level at which all other downward stream batches are truncated.

$$\epsilon^b = c + n^{arr} + 1 \leq j \leq L - 1 - n^{serv} - n^{kill} = \epsilon^t$$

We call levels that lie below the repeating region the *filling* region and those that lie above[2] the *blocking* region. Figure 3.1 illustrates the location of the regions for the simple two-state modulated queue we have considered earlier. In most applications

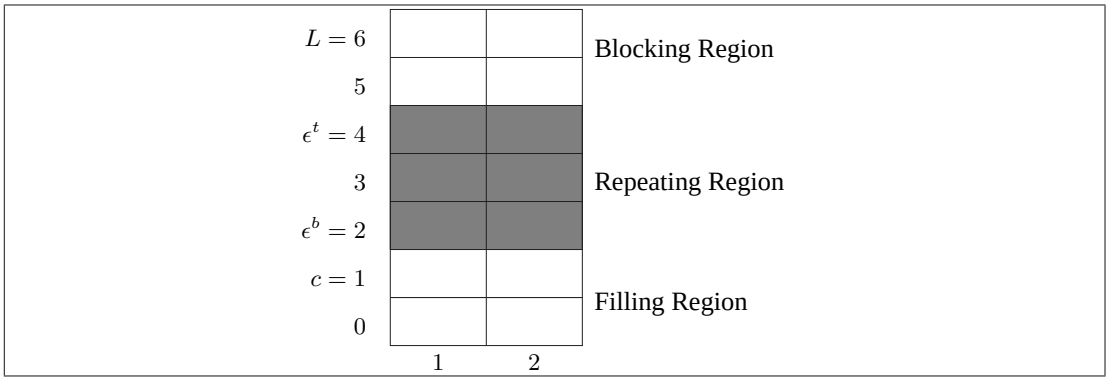


Figure 3.1: Queue Regions for an unbatched finite queue with one arrival and one service stream.

the repeating region will cover the majority of the state space of the queue. Whenever this is not the case (such as when the queue is very short), there is no computational advantage in employing the matrix methods discussed below. In this instance it is advisable to employ direct solution methods like Gaussian-elimination.

3.3 Repeating region solution

With localised balance equations available, we can use two related, but seemingly distinct methods to solve for repeating regions in our queues: Modern derivatives of the Matrix Geometric method and the Spectral Expansion method. The methods

[2] finite capacity queues only

essentially differ in the manner in which the relationship between adjacent levels is represented.

The Classic Matrix Geometric (MG) Method, first studied in [Neu81], does this using a matrix R, whose linear operations when applied to a SOP vector within the repeating region are assumed to be sufficient to yield the SOP of the next level.

Spectral expansion, originally introduced by Chakka and Mitrani in [Cha95], on the other hand aims to calculate repeating region SOPs via linear combinations of appropriately scaled eigenvectors inherent in the recurrence relation.

The questions of relative stability and efficiency of the two methods is vexed, as it pivots on the development of new computational routines. For example, in 1981 the MG method was initially based on an iterative Simple Substitution (SS) [Neu81] method which took many iterations to converge, especially for heavy-load systems. A major improvement was achieved in 1993 by the introduction of the Logarithmic Reduction (LR) algorithm [LatRam93] by Latouche and Ramaswami. The LR algorithm is computationally more expensive at each iteration, but many fewer iteration steps are required, even for systems whose load approaches unity. Subsequent improvements [NaoKri+97] in 1997 by Naoumov at al. have been more incremental, yet still worthwhile.

The Spectral Expansion method [Cha95], proposed in 1995, depends heavily on available eigenvalue and eigenvector packages [WilRei71], many of which are very mature and reliable [Smi+76].

The various comparisons between MG and SE that have been done in the past usually favour SE. For example, in Haverkort and Ost's review [HavOst97] it is concluded that

"...the spectral expansion method is favourable in all cases, especially when more heavily loaded systems are studied and when batch arrivals (or departures) are included in the model".

Mitrani and Chakka make very similar observations [MitCha95], based on a comparison with Neuts' original method [Neu81] in where they state that

"Spectral Expansion offers considerable advantages in efficiency, without undue penalties in terms of numerical stability. The speed gains are especially pronounced, and indeed appear to be unbounded, when the system approaches saturation".

Nevertheless, Matrix Geometric Methods enjoy great popularity and we will provide steady-state solution mechanisms for both MG and SE Methods.

We will provide examples in which each of the methods experience problems in obtaining a solution. In this process, we will later show[3] that there is significant overlap between the two methods, which can be used to provide faster and more stable solutions in many cases. We now outline the two methods, starting with Spectral expansion, and identify issues that arise from their application to our localised balance equations.

3.4 Spectral expansion solution

The spectral expansion solution to the linear homogeneous matrix equation (3.1) is the sum of geometric series provided by the eigenvalues ξ_i of the characteristic matrix

$$\underline{\mathbf{Q}}(\xi) = \sum_{k=0}^{n^{arr}+n^{serv}+n^{kill}} \xi^k \underline{\mathbf{Q}}_k \tag{3.3}$$

found by setting $\det \underline{\mathbf{Q}}(\xi) = 0$. Each of the eigenvalues ξ_i is then used to determine its corresponding eigenvector $\overrightarrow{\boldsymbol{\psi}}_i$ from the set of N linear equations $\overrightarrow{\boldsymbol{\psi}}_i \underline{\mathbf{Q}}(\xi_i) = 0$.

Let there be n^* eigenvalue/eigenvector pairs. Each of these defines a basis function component, and by summing the ensemble with each component scaled appropriately, we can satisfy the boundary conditions defined by the balance equations of the levels within the filling and blocking regions as well as the normalisation constraint.

At queue lengths j falling under the repeating region balance equation, we have:

$$\overrightarrow{\mathbf{v}}_j = \textstyle\sum_{i=1}^{n^*} \alpha_i \xi_i^j \overrightarrow{\boldsymbol{\psi}}_i \quad \epsilon^b \leq j \leq \epsilon^t \tag{3.4}$$

The coefficients α_i are free variables to be constrained by boundary conditions. These boundary conditions are imposed by the inclusion of balance equations which include

[3] Mitrani already mentioned a special case in [MitCha95]

both explicit $\overrightarrow{\mathbf{v}}_j$ vectors outside the repeating region, and vectors defined by the eigensystem (3.4).

The choice of raising the eigenvalue to the power j, the queue level, is somewhat arbitrary: the scaling due to any constant offset, $j + X$, could be absorbed into the associated α value.

When the queue is unbounded, any eigenvalues of magnitude greater than or equal to 1 (*i.e.* lying on or outside the unit disk in the Argand diagram) must take zero coefficients, as their infinite sum does not converge, and hence the solutions cannot be normalised. Eigenvectors with zero eigenvalue can also not contribute to the solution as, in a geometric series.

3.4.1 Number of eigenvalues

The number of eigenvalues n^* pertaining to this system is dictated by the polynomial order of the determinant of the characteristic matrix in ξ as well as the size of the matrices involved. Thus, before any of the considerations concerning zero eigenvalues and those situated outside the unit disc, we have

$$n^* = (n^{arr} + n^{serv} + n^{kill})N$$

Zero Eigenvalues

As described in [ThoZat+03], zero eigenvalues appear due to the presence of *degenerate* streams, that is streams that duplicate the behaviour of one another. An unbatched stream (*i.e.* with batch parameter 0) does not introduce unbounded dependencies throughout the queue. Level transitions are limited to one up or one down, so that localisation is not needed. However, if the localisation procedure is applied, a zero eigenvalue is introduced. Another source of degeneracy is given by two or more streams that transition in the same direction and exhibit the same batch size distribution parameter. These are merged by adding their transition rates and leaving the batch parameter unchanged. Of particular note is that within the repeating region, the transition behaviour of service and killing streams is indistinguishable such that even these

two distinct types of streams can be merged. When localisation is performed without the merging, a zero eigenvalue is added for each modulation state where equal batch parameters occur to give the total number of degenerate streams (and associated zero eigenvalues of equal number). We write n_m^{arr} as the number of distinct arrival streams in modulation state m and similarly for the service and negative customer streams. The total number of non-zero eigenvalues are then written as

$$n^* = \sum_{m=1}^{N} (n_m^{arr} + n_m^{serv} + n_m^{kill})$$

This is the effective number of eigenvalues used for finite queues. When using spectral expansion to solve an infinite queue that is not overloaded we need to additionally drop the eigenvalues that lie outside the unit disc. For a small problem, with only two modulation states and one arrival stream it can be shown by direct solution for the eigenvalues that there are (in the absence of degenerate streams whose zero-eigenvalues have already been dropped) a total of 2×1. This result is difficult to generalise due to the large number of parameters occurring in quintic and higher order polynomial, and we conjecture that in general problems with N modulation states and no degenerate streams, the number of eigenvalues within the unit disc is $N \times n^{arr}$, so that for non-overloaded infinite queues the number of eigenvalues in the unit disc is

$$n^* = \sum_{m=1}^{N} n_m^{arr}$$

Complex Eigenvalues

Normally, every one of the n^* eigenvalue/eigenvector pairs that have not been dropped corresponds to one of the free variables α_i that the SE method uses to constrain the repeating region. An exception to this occurs when an eigenvalue is complex. The characteristic equation derived from (3.3) has only real coefficients so that any complex eigenvalues appear in complex conjugate pairs. The geometric series of each of these traces a spiral in the Argand diagram. Since the solution must be real, the imaginary parts must cancel out. As a consequence, the corresponding α coefficients must also be mutually complex conjugate, reducing the degrees of freedom by one for every complex conjugate eigenvalue pair encountered. We give a simple example

queue where complex eigenvalues can occur. It is infinite ($L = \infty$), with one processor ($c = 1$), three modulation states ($N = 3$) and no geometric batches nor negative customers. We use $\underline{\mathbf{\Lambda}} = \underline{\mathbf{I}}$ and $\underline{\mathbf{M}} = \underline{\mathbf{diag}}(2, 3/2, 0)$. The modulation process we use causes the three modulation states always being visited in the order $1 \mapsto 2 \mapsto 3 \mapsto 1$.

$$\underline{\mathbf{Q}} = k \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$

k is used to control the *speed* of the modulation process. Large values of k give fast modulation, small k lead to slow modulation. At the same time the load of the queue is unaffected by the value of k. Figures 3.2 and 3.3 show how real and imaginary parts of two of the three eigenvalues found in the eigensystem of the repeating region vary with the modulation speed. For slow modulation the two eigenvalues are real and distinct, whereas faster modulation gives a complex conjugate pair of eigenvalues. This transition occurs at approximatively $k = 0.27$, where $\xi_1 = \xi_2$. The exact value for k cannot be easily calculated as it requires, for this example, the solution of a quintic polynomial.

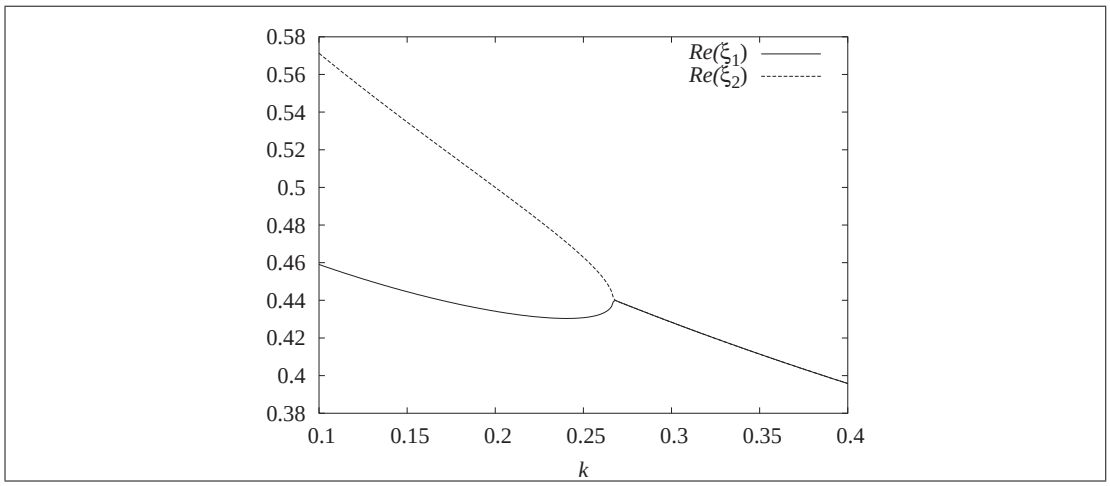


Figure 3.2: Real parts of two selected eigenvalues, as the modulation speed k changes.

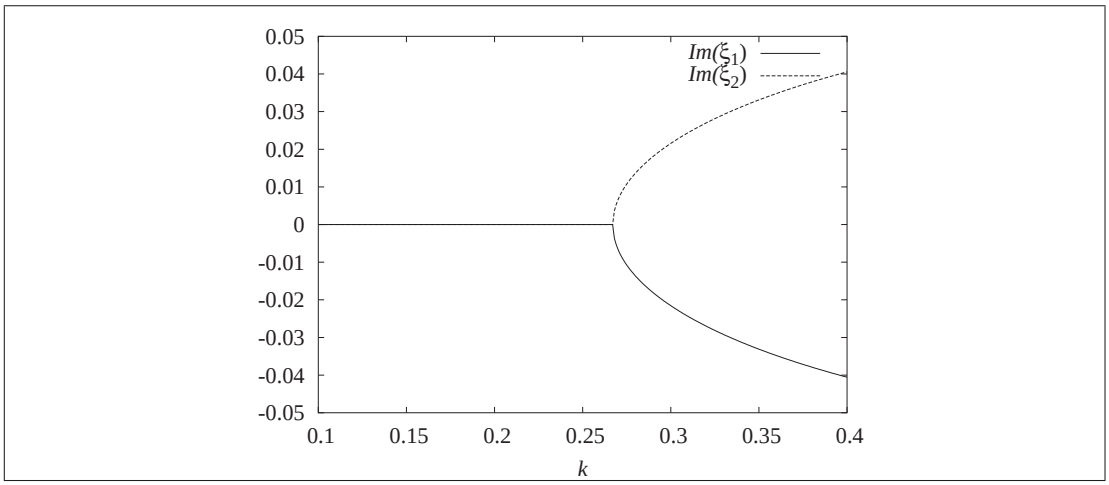


Figure 3.3: Imaginary parts of two selected eigenvalues, as modulation speed k changes.

3.4.2 Limitations of the SE method

The practical deployment of the spectral expansion method has not changed significantly since early treatments, for example by Mitrani in 1995 [MitCha95]. The main desirable improvements would be in terms of efficiency, as solutions are generally accurate. The only fundamental issue that needs to be addressed occurs in the vicinity of *exact saturation*.

A queue is said to be exactly saturated, when customers arrive with the exact overall rate at which they depart (due to service and negative customers). Essentially, this implies that

$$\|\overrightarrow{\pi}(c\underline{\mathbf{M}} + \underline{\mathbf{K}})\|_1 = \|\overrightarrow{\pi}\underline{\mathbf{\Lambda}}\|_1$$

where $\overrightarrow{\pi}$ is the equilibrium solution vector of the modulation chain, *i.e.* it satisfies $\overrightarrow{\pi}\underline{\mathbf{Q}} = \overrightarrow{\mathbf{0}}$. When a queue approaches this exact saturation, a non-unit eigenvalue, smaller in magnitude than 1, approaches the absolute value 1. There is always another unit eigenvalue (associated with eigenvector $\overrightarrow{\pi}$).

To see why this is the case, consider the example repeating region, localised balance

equation (3.2).

$$\begin{aligned}\overrightarrow{\mathbf{v}}_{j-1}(\underline{\mathbf{M}\boldsymbol{\Theta}} - \underline{\mathbf{Q}\boldsymbol{\Theta}} + \underline{\boldsymbol{\Lambda}}) \\ +\overrightarrow{\mathbf{v}}_{j}(\underline{\mathbf{Q}}(\underline{\boldsymbol{\Theta}\boldsymbol{\Phi}} + \underline{\mathbf{I}}) - \underline{\mathbf{M}}(\underline{\boldsymbol{\Theta}} + \underline{\mathbf{I}}) - \underline{\boldsymbol{\Lambda}}(\underline{\boldsymbol{\Phi}} + \underline{\mathbf{I}})) \\ +\underline{\mathbf{v}}_{j+1}(\underline{\mathbf{M}} - \underline{\mathbf{Q}\boldsymbol{\Phi}} + \underline{\boldsymbol{\Lambda}\boldsymbol{\Phi}}) = \overrightarrow{\mathbf{0}}\end{aligned}$$

To verify whether there exists an eigenvector with eigenvalue 1 we write

$$\overrightarrow{\mathbf{v}}_j = 1^j\overrightarrow{\boldsymbol{\psi}} = \overrightarrow{\boldsymbol{\psi}}$$

$$\begin{aligned}\overrightarrow{\boldsymbol{\psi}}(\underline{\mathbf{M}\boldsymbol{\Theta}} - \underline{\mathbf{Q}\boldsymbol{\Theta}} + \underline{\boldsymbol{\Lambda}}) \\ +\overrightarrow{\boldsymbol{\psi}}(\underline{\mathbf{Q}}(\underline{\boldsymbol{\Theta}\boldsymbol{\Phi}} + \underline{\mathbf{I}}) - \underline{\mathbf{M}}(\underline{\boldsymbol{\Theta}} + \underline{\mathbf{I}}) - \underline{\boldsymbol{\Lambda}}(\underline{\boldsymbol{\Phi}} + \underline{\mathbf{I}})) \\ +\overrightarrow{\boldsymbol{\psi}}(\underline{\mathbf{M}} - \underline{\mathbf{Q}\boldsymbol{\Phi}} + \underline{\boldsymbol{\Lambda}\boldsymbol{\Phi}}) = \overrightarrow{\mathbf{0}}\end{aligned}$$

After simplification this equation transforms to

$$\overrightarrow{\boldsymbol{\psi}}\underline{\mathbf{Q}} = \overrightarrow{\mathbf{0}}$$

so that $\overrightarrow{\boldsymbol{\psi}} = \overrightarrow{\pi}$ is indeed an eigenvector of the recurrence relation with eigenvalue 1.

The proximity of these two eigenvalues at 1 introduces instability in the solution and eventually, at the exact point of saturation (which of course is only important for a finite queue), the eigenvalue 1 occurs with multiplicity two, but there exists only one eigenvector for this eigenvalue. As a simple illustration of how this can happen we consider the following matrix, that has a double eigenvalue at 2, but only one corresponding left eigenvector $(1, 0)$

$$\begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix}$$

The issue of encountering similarly behaved matrices in our queueing problems prompted the possibility of extending SE using generalised eigenvectors. The outcome of our analysis clarifies the method's behaviour in circumstances close to exact saturation of the queue, and allows us to prove the completeness of the modified spectral expansion solution.

3.5 Generalised Spectral Expansion Method

In addition to the numerical instabilities that occur due to the proximity of eigenvalues to each other, one eigenvector disappears at the exact saturation point. This discontinuity in the number of solution (eigen) vectors causes conditioning problems in the linear system used to evaluate the free variables α_i. Another situation in which this behaviour might occur has already been shown in the discussion of complex eigenvalues where, for certain parameters there is a double eigenvalue.

We extend the analysis of the characteristic matrix $\underline{\mathbf{Q}}(\xi)$ using generalised eigenvectors which removes the discontinuity and allows us to prove that our generalised spectral solution is complete. In addition, the analysis also specifies the locus of problems, in terms of associated eigenvalues, which are not amenable to solution using MG methods. Our work with geometric batches revealed these shortcomings, and we now briefly outline the stages of development.

Consider a recurrence relation of order M.

$$\sum_{k=0}^{M} \overrightarrow{\mathbf{v}}_{j+k} \underline{\mathbf{Q}}_k = \overrightarrow{\mathbf{0}}^T \qquad \epsilon^b \leq j \leq \epsilon^t \tag{3.5}$$

where ϵ^b and ϵ^t are the bottom and top levels at which this relation holds.

If $\underline{\mathbf{Q}}_M$ is non-singular, we can re-write this as

$$\overrightarrow{\mathbf{v}}_{j+M} = -\sum_{k=0}^{M-1} \overrightarrow{\mathbf{v}}_{j+k} \underline{\mathbf{Q}}_k \underline{\mathbf{Q}}_M^{-1}$$

or, equivalently

$$\overrightarrow{\mathbf{v}}_{j+M} = [\overrightarrow{\mathbf{v}}_j, \ldots, \overrightarrow{\mathbf{v}}_{j+M-1}]\underline{\mathbf{B}} \tag{3.6}$$

where

$$\underline{\mathbf{B}} = \begin{pmatrix} -\underline{\mathbf{Q}}_0 \underline{\mathbf{Q}}_M^{-1} \\ \vdots \\ -\underline{\mathbf{Q}}_{M-1} \underline{\mathbf{Q}}_M^{-1} \end{pmatrix}$$

When $\underline{\mathbf{Q}}_M$ is singular, but $\underline{\mathbf{Q}}_0$ is not, we can similarly write

$$\overrightarrow{\mathbf{v}}_j = -\sum_{k=1}^{M} \overrightarrow{\mathbf{v}}_{j+k} \underline{\mathbf{Q}}_k \underline{\mathbf{Q}}_0^{-1}$$

and continue in an analogous manner. When both $\underline{\mathbf{Q}}_M$ and $\underline{\mathbf{Q}}_0$ are singular, we can use a transformation of the variable $\xi \mapsto \frac{1+\xi}{1-\xi}$ in (3.3). Such a transformation has already been mentioned in [MitCha95], where it was used on a problem of size $M = 2$. The matrix polynomial that results from this transformation has a modified matrix $\underline{\mathbf{Q}}'_0$ that is always non-singular so that the above mentioned methods can be used for the solution process. Naturally, once eigenvalues have been found, they need to be transformed back via $\xi \mapsto \frac{\xi-1}{\xi+1}$.

To enable the use of standard matrix operations and measures that require square matrices, we add a set of $M-1$ vector equations of the form $\overrightarrow{\mathbf{v}}_{j+k} = \overrightarrow{\mathbf{v}}_{j+k}$ to the one in (3.6) that do not change the solution set of the original matrix equation as follows.

$$[\overrightarrow{\mathbf{v}}_{j+1}, \overrightarrow{\mathbf{v}}_{j+2}, \ldots, \overrightarrow{\mathbf{v}}_{j+M}] = [\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_{j+1}, \ldots, \overrightarrow{\mathbf{v}}_{j+M-1}]\underline{\mathbf{A}}$$

where

$$\underline{\mathbf{A}} = \begin{bmatrix} \underline{\mathbf{0}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} & -\underline{\mathbf{Q}}_0\underline{\mathbf{Q}}_M^{-1} \\ \underline{\mathbf{I}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} & -\underline{\mathbf{Q}}_1\underline{\mathbf{Q}}_M^{-1} \\ \underline{\mathbf{0}} & \underline{\mathbf{I}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} & -\underline{\mathbf{Q}}_2\underline{\mathbf{Q}}_M^{-1} \\ \vdots & \vdots & \vdots & & & \vdots \\ \underline{\mathbf{0}} & \underline{\mathbf{0}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{I}} & -\underline{\mathbf{Q}}_{M-1}\underline{\mathbf{Q}}_M^{-1} \end{bmatrix} \tag{3.7}$$

We now define a new, compound, $1 \times NM$ solution vector

$$\overrightarrow{\mathbf{w}}_j = [\overrightarrow{\mathbf{v}}_j, \overrightarrow{\mathbf{v}}_{j+1}, \ldots, \overrightarrow{\mathbf{v}}_{j+M-1}]$$

That gives

$$\overrightarrow{\mathbf{w}}_{j+1} = \overrightarrow{\mathbf{w}}_j\underline{\mathbf{A}} \Leftrightarrow \overrightarrow{\mathbf{w}}_{j+i} = \overrightarrow{\mathbf{w}}_j\underline{\mathbf{A}}^i \quad \epsilon^b \leq j \leq j+i \leq \epsilon^t \tag{3.8}$$

We will now study the spectrum of $\underline{\mathbf{A}}$ in detail, in order to give a full account of the solution space for the queue length distribution vector $\overrightarrow{\mathbf{v}}_j$.

3.5.1 Jordan form

Every square matrix $\underline{\mathbf{A}}$ is similar[4] to a unique Canonical Jordan Form $\underline{\mathbf{J}}$, *i.e.*

$$\underline{\mathbf{A}} = \underline{\mathbf{V}}^{-1}\underline{\mathbf{J}}\,\underline{\mathbf{V}}$$

where the columns of $\underline{\mathbf{V}}$ are given by the linearly independent eigenvectors of $\underline{\mathbf{A}}$ and

$$\underline{\mathbf{J}} = \underline{\mathbf{diag}}(\underline{\mathbf{J}}_1, \underline{\mathbf{J}}_2, \ldots, \underline{\mathbf{J}}_r) \tag{3.9}$$

is block-diagonal, with the i-th block $\underline{\mathbf{J}}_i$ associated with eigenvalue λ_i given by

$$\underline{\mathbf{J}}_i = \begin{bmatrix} \lambda_i & 0 & 0 & 0 & \cdots & 0 \\ 1 & \lambda_i & 0 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 & \lambda_i \end{bmatrix}$$

In the literature (*e.g.* [Pea65, Dei91]), Jordan forms are traditionally used with respect to right multiplication of vectors. The solution for Markov modulated queues is invariably carried out using left-multiplication. Thus we re-cast the use of Jordan forms to use left-vectors, which requires transposition of the matrix system.

Each Jordan block corresponds to exactly one eigenvalue/eigenvector pair $(\lambda_i, \overrightarrow{\boldsymbol{\psi}}_i)$ of $\underline{\mathbf{A}}$. If there are NM Jordan blocks, *i.e.* all blocks are of size 1×1, it follows that $\underline{\mathbf{A}}$ is diagonalisable,*i.e.* similar to a diagonal matrix, and that it has NM linearly independent eigenvectors $\overrightarrow{\boldsymbol{\psi}}_i$.

3.5.2 Generalised eigenvectors

If there are fewer than NM Jordan blocks, r say, there will only be r linearly independent eigenvectors, $\overrightarrow{\boldsymbol{\psi}}_i$. In this case, for each Jordan block $\underline{\mathbf{J}}_i$ of size $s_i \geq 2$, $1 \leq i \leq r$,

[4] A matrix $\underline{\mathbf{A}}$ is said to be similar to a matrix $\underline{\mathbf{B}}$ if there exists a non-singular matrix $\underline{\mathbf{V}}$ such that $\underline{\mathbf{A}} = \underline{\mathbf{V}}^{-1}\underline{\mathbf{B}}\,\underline{\mathbf{V}}$. Similar matrices have the same set of eigenvalues and represent the same linear transformation in a different basis.

we write $\overrightarrow{\psi}_{i,1} = \overrightarrow{\psi}_i$ and generate a set of *generalised eigenvectors* $\left\{\overrightarrow{\psi}_{i,2}, \cdots, \overrightarrow{\psi}_{i,s_i}\right\}$ by using the recurrence relation

$$\overrightarrow{\psi}_{i,k}(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) = \overrightarrow{\psi}_{i,k-1} \qquad k \geq 2 \tag{3.10}$$

The Matrix $\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}$ is singular with a kernel[5] dimension s_i and hence at every stage there are multiple, linearly independent solutions when calculating $\overrightarrow{\psi}_{i,k}$ using the above formula. Indeed, if $\overrightarrow{\psi}_{i,k}$ satisfies (3.10), then $\overrightarrow{\psi}_{i,k} + a\overrightarrow{\psi}_{i,1}$ does so as well for any value $a \in \mathbb{R}$, because

$$(\overrightarrow{\psi}_{i,k} + a\overrightarrow{\psi}_{i,1})(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) = \overrightarrow{\psi}_{i,k-1} + a\lambda_i \overrightarrow{\psi}_{i,1} - a\lambda_i \overrightarrow{\psi}_{i,1} = \overrightarrow{\psi}_{i,k-1} \tag{3.11}$$

Any value of a can be used however, as the following Lemma demonstrates.

Lemma 1. *Any set* $\left\{\overrightarrow{\psi}_{i,1}, \overrightarrow{\psi}_{i,2}, \ldots, \overrightarrow{\psi}_{i,s_i}\right\}$*, calculated using the recurrence relation* $\overrightarrow{\psi}_{i,k}(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) = \overrightarrow{\psi}_{i,k-1}$ *is linearly independent.*

Proof by induction

Let $P(n)$ be the statement that the set $\left\{\overrightarrow{\psi}_{i,1}, \overrightarrow{\psi}_{i,2}, \ldots, \overrightarrow{\psi}_{i,n}\right\}$ is linearly independent for $n \geq 1$.

$P(1)$ is satisfied since the eigenvector cannot be the zero vector. We now assume that $P(n)$ holds for $1 \leq n \leq s_i$.

Assume that the set $\left\{\overrightarrow{\psi}_{i,1}, \overrightarrow{\psi}_{i,2}, \ldots, \overrightarrow{\psi}_{i,n+1}\right\}$ is linearly dependent, *i.e.* $\overrightarrow{\psi}_{i,n+1} = \sum_{l=1}^{n} g_l \overrightarrow{\psi}_{i,l}$ for some constants g_l. From the recurrence relation,

$$\begin{aligned}
\overrightarrow{\psi}_{i,n} &= \overrightarrow{\psi}_{i,n+1}(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) \\
&= \textstyle\sum_{l=1}^{n} g_l \overrightarrow{\psi}_{i,l}(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) \\
&= \textstyle\sum_{l=2}^{n} g_l \overrightarrow{\psi}_{i,l}(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) + g_1 \overrightarrow{\psi}_{i,1}(\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}) \\
&= \textstyle\sum_{l=2}^{n} g_l \overrightarrow{\psi}_{i,l-1} + g_1 \lambda_i \overrightarrow{\psi}_{i,1} - g_1 \lambda_i \overrightarrow{\psi}_{i,1} \\
&= \textstyle\sum_{l=1}^{n-1} g_{l+1} \overrightarrow{\psi}_{i,l}
\end{aligned}$$

[5] $\underline{\mathbf{A}}$ could have other Jordan blocks with the same eigenvalue λ_k, which make the kernel of $\underline{\mathbf{A}} - \lambda_i \underline{\mathbf{I}}$ even larger, but those blocks would have a different, linearly independent set of eigenvectors associated with them

This contradicts the induction hypothesis $P(n)$ and hence our set must be linearly independent which proves $P(n+1)$■

The vectors then satisfy

$$\begin{aligned}\overrightarrow{\boldsymbol{\psi}}_{i,1}\underline{\mathbf{A}} &= \lambda_i \overrightarrow{\boldsymbol{\psi}}_{i,1} && k = 1\\ \overrightarrow{\boldsymbol{\psi}}_{i,k}\underline{\mathbf{A}} &= \lambda_i \overrightarrow{\boldsymbol{\psi}}_{i,k} + \overrightarrow{\boldsymbol{\psi}}_{i,k-1} && 2 \leq k \leq s_i\end{aligned}$$

When applied to all Jordan blocks, we derive sets of eigenvectors and generalised eigenvectors, the union of which is linearly independent and of size NM. Vectors from distinct Jordan blocks are necessarily linearly independent, as the following proposition demonstrates.

Proposition 1. *Eigenvector and generalised eigenvectors of* $\underline{\mathbf{A}}$ *from distinct Jordan blocks are linearly independent.*

Proof

$\underline{\mathbf{A}}$ and $\underline{\mathbf{J}}$ represent the same linear transformation, but use a different set of basis vectors. $\underline{\mathbf{J}}$ is block-diagonal, so that clearly the eigenvectors and generalised eigenvectors from distinct Jordan blocks are linearly independent. ■

3.5.3 Linear map

The matrix $\underline{\mathbf{A}}$ represents a linear map $\mathbb{R}^{NM} \mapsto \mathbb{R}^{NM}$ on vectors. Being linear, it is fully specified by the transformations it effects on any set of vectors that span the underlying space. $\underline{\mathbf{A}}$ is $NM \times NM$ and we have found, in the form of eigenvectors $\overrightarrow{\boldsymbol{\psi}}_{k,1}$ (and generalised eigenvectors $\overrightarrow{\boldsymbol{\psi}}_{k,i}$), a set of NM vectors that are linearly independent. We therefore have a full basis and can rewrite the solution vector at the bottom of the described region (level ϵ^b) (3.8) as a linear combination of the components of the basis.

$$\overrightarrow{\mathbf{w}}_{\epsilon^b} = \sum_{k=1}^{r}\left[\sum_{l=1}^{s_k} \alpha_{k,l}\overrightarrow{\boldsymbol{\psi}}_{k,l}\right]$$

Equilibrium solutions at levels in the range ϵ^b through ϵ^t are uniquely determined by the repeated application of the linear map $\underline{\mathbf{A}}$. The evaluation of the powers of the matrix $\underline{\mathbf{A}}$,

$$\underline{\mathbf{A}}^j = \underline{\mathbf{V}}^{-1}\underline{\mathbf{J}}^j\underline{\mathbf{V}}$$

is made efficient through the evaluation of powers of a block-diagonal matrix (3.9), which is given by:

$$\underline{\mathbf{J}}^j = \underline{\mathbf{diag}}(\underline{\mathbf{J}}_1^j, \underline{\mathbf{J}}_2^j, \ldots, \underline{\mathbf{J}}_r^j)$$

To calculate $\underline{\mathbf{J}}_i^j$, we write $\underline{\mathbf{J}}_i = \lambda_i\underline{\mathbf{I}} + \underline{\mathbf{N}}$, where $\underline{\mathbf{N}}$ has 1's below its diagonal and 0's everywhere else *i.e.* it is a banded matrix with 1's on the first lower diagonal and zeros otherwise. Consequently, $\underline{\mathbf{N}}^j$ is also a banded matrix, with 1's on its j^{th} lower diagonal and zeros otherwise.

Now,

$$\underline{\mathbf{J}}_i^j = (\lambda_i\underline{\mathbf{I}} + \underline{\mathbf{N}})^j = \sum_{m=0}^{j} \begin{pmatrix} j \\ m \end{pmatrix} \lambda_i^{j-m}\underline{\mathbf{N}}^m$$

Hence, the powers of the Jordan blocks $\underline{\mathbf{J_i}}$ are lower triangular and are given by:

$$\underline{\mathbf{J_i^j}} = \begin{bmatrix} \lambda_i^j & 0 & \cdots & 0 & 0 \\ \begin{pmatrix} j \\ 1 \end{pmatrix}\lambda_i^{j-1} & \lambda_i^j & \cdots & 0 & 0 \\ \begin{pmatrix} j \\ 2 \end{pmatrix}\lambda_i^{j-2} & \begin{pmatrix} j \\ 1 \end{pmatrix}\lambda_i^{j-1} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \lambda_i^j & 0 \\ \begin{pmatrix} j \\ s_i - 1 \end{pmatrix}\lambda_i^{j-s_i+1} & \begin{pmatrix} j \\ s_i - 2 \end{pmatrix}\lambda_i^{j-s_i+2} & \cdots & \begin{pmatrix} j \\ 1 \end{pmatrix}\lambda_i^{j-1} & \lambda_i^j \end{bmatrix}$$

using the convention that $\begin{pmatrix} j \\ k \end{pmatrix} = 0$ whenever $j < k$.

3.5.4 Completeness of generalised spectral expansion

Assembling the preceding results,

$$\overrightarrow{\mathbf{w}}_{\epsilon^b+j} = \overrightarrow{\mathbf{w}}_{\epsilon^b}\underline{\mathbf{A}}^j = \overrightarrow{\mathbf{w}}_{\epsilon^b}\underline{\mathbf{V}}^{-1}\underline{\mathbf{J}}^j\underline{\mathbf{V}}$$

Proposition 2.

$$\overrightarrow{\mathbf{w}}_{\epsilon^b+j} = \sum_{k=1}^{r}\left[\sum_{l=1}^{s_k}\alpha_{k,l}\left[\sum_{m=0}^{l-1}\begin{pmatrix} j \\ m \end{pmatrix}\lambda_k^{j-m}\overrightarrow{\boldsymbol{\psi}}_{k,l-m}\right]\right]$$

represents the complete solution to the recurrence relation (3.1).

Proof

$\underline{\mathbf{V}}$ is a matrix whose columns are given by the eigenvectors and generalised eigenvectors of $\underline{\mathbf{A}}$. Hence,

$$\begin{aligned} &\overrightarrow{\mathbf{e}}_n.\underline{\mathbf{V}} = \overrightarrow{\boldsymbol{\psi}}_{n,1} \quad \text{if all Jordan blocks are of size } 1 \\ &\overrightarrow{\mathbf{e}}_n.\underline{\mathbf{V}} = \overrightarrow{\boldsymbol{\psi}}_{k,l} \quad \text{otherwise, for some values } k \text{ and } l \end{aligned} \tag{3.12}$$

For the multiplication with its inverse, $\underline{\mathbf{V}}^{-1}$, it follows

$$\begin{aligned} &\overrightarrow{\boldsymbol{\psi}}_{n,1}.\underline{\mathbf{V}}^{-1} = \overrightarrow{\mathbf{e}}_n \quad \text{if all Jordan blocks are of size } 1 \\ &\overrightarrow{\boldsymbol{\psi}}_{k,l}.\underline{\mathbf{V}}^{-1} = \overrightarrow{\mathbf{e}}_n \quad \text{otherwise, for some value of } n \end{aligned} \tag{3.13}$$

Case 1: All Jordan block of size 1

We follow one of the eigenvector components of $\overrightarrow{\mathbf{w}}_{\epsilon^b}$, $\alpha_{n,1}\overrightarrow{\boldsymbol{\psi}}_{n,1}$ say, as it is multiplied by $\underline{\mathbf{A}}^j = \underline{\mathbf{V}}^{-1}\underline{\mathbf{J}}^j\underline{\mathbf{V}}$.

$$\begin{aligned} \alpha_{n,1}\overrightarrow{\boldsymbol{\psi}}_{n,1}\underline{\mathbf{A}}^j &= \alpha_{n,1}\left(\overrightarrow{\boldsymbol{\psi}}_{n,1}\underline{\mathbf{V}}^{-1}\right)\underline{\mathbf{J}}^j\underline{\mathbf{V}} \\ &= \alpha_{n,1}\left(\overrightarrow{\mathbf{e}}_n\underline{\mathbf{J}}^j\right)\underline{\mathbf{V}} \\ &= \alpha_{n,1}\lambda_n^j\overrightarrow{\mathbf{e}}_n\underline{\mathbf{V}} \\ &= \alpha_{n,1}\lambda_n^j\overrightarrow{\boldsymbol{\psi}}_{n,1} \end{aligned}$$

Summing over all the eigenvector components now gives the familiar spectral expansion method expression, which agrees with the result stated in the proposition, when $s_k = 1$ and $r = NM$.

$$\overrightarrow{\mathbf{w}}_{\epsilon^b+j} = \sum_{n=1}^{NM}\alpha_{n,1}\lambda_n^j\overrightarrow{\boldsymbol{\psi}}_{n,1}$$

This also means that whenever a complete set of NM eigenvectors can be found, which is always the case if all eigenvalues are distinct, Spectral Expansion, as a subset of GSE, is complete also.

Case 2: At least one Jordan Block larger than 1

The majority of cases are covered by case 1, as multiple eigenvalues require very specific relationships between queue parameters. The presence of a double eigenvalue as shown in figures 3.2 and 3.3 is not sufficient. Indeed the example shown there exhibits two distinct Jordan blocks of size one, where each of the double eigenvalue retains its unique eigenvector.

Subspaces associated with distinct Jordan blocks are disjoint. The following argument therefore applies to each one separately. For simplicity, we observe the behaviour of the eigenvectors and generalised eigenvectors of a single, isolated Jordan block. Let the Jordan block be the k^{th} of the matrix $\underline{\mathbf{A}}$. It is of size s_k, with eigenvalue λ_k, eigenvector $\overrightarrow{\boldsymbol{\psi}}_{k,1}$ and generalised eigenvectors $\overrightarrow{\boldsymbol{\psi}}_{k,2}, \ldots, \overrightarrow{\boldsymbol{\psi}}_{k,s_k}$.

For the eigenvector component, $\alpha_{k,1}\overrightarrow{\boldsymbol{\psi}}_{k,1}$, we have, as in case 1

$$\alpha_{k,1}\overrightarrow{\boldsymbol{\psi}}_{k,1}\underline{\mathbf{A}}^j = \alpha_{k,1}\lambda_k^j\overrightarrow{\boldsymbol{\psi}}_{k,1}$$

When the generalised eigenvector component $\alpha_{k,l}\overrightarrow{\boldsymbol{\psi}}_{k,l}$ is used

$$\begin{aligned}
\alpha_{k,l}\overrightarrow{\boldsymbol{\psi}}_{k,l}\underline{\mathbf{A}}^j &= \alpha_{k,l}\left(\overrightarrow{\boldsymbol{\psi}}_{k,l}\underline{\mathbf{V}}^{-1}\right)\underline{\mathbf{J}}^j\underline{\mathbf{V}} \\
&= \alpha_{k,l}\left(\overrightarrow{\mathbf{e}}_n\underline{\mathbf{J}}^j\right)\underline{\mathbf{V}} \\
&= \alpha_{k,l}\sum_{m=0}^{l-1}\lambda_n^{j-m}\begin{pmatrix} j \\ m \end{pmatrix}\overrightarrow{\mathbf{e}}_{n-m}\underline{\mathbf{V}} \\
&= \alpha_{k,l}\sum_{m=0}^{l-1}\lambda_n^{j-m}\begin{pmatrix} j \\ m \end{pmatrix}\overrightarrow{\boldsymbol{\psi}}_{k,l-m}
\end{aligned}$$

We now sum all the components within this Jordan block

$$\sum_{l=1}^{s_k}\alpha_{k,l}\overrightarrow{\boldsymbol{\psi}}_{k,l}\underline{\mathbf{A}}^j = \sum_{l=1}^{s_k}\alpha_{k,l}\sum_{m=0}^{l-1}\lambda_n^{j-m}\begin{pmatrix} j \\ m \end{pmatrix}\overrightarrow{\boldsymbol{\psi}}_{k,l-m}$$

Additionally summing over all Jordan blocks $1 \leq k \leq r$ gives the required result. ■

3.5.5 Efficient reduction to $\mathbb{R}^N$ space

The original probability vectors $\overrightarrow{\mathbf{v}}_j$ can be trivially reconstructed from $\overrightarrow{\mathbf{w}}_j$ by means of a $\mathbb{R}^{NM} \mapsto \mathbb{R}^N$ projection matrix

$$\underline{\mathbf{P}} = \begin{pmatrix} \underline{\mathbf{I}}_N & \underline{\mathbf{0}}_N & \cdots & \underline{\mathbf{0}}_N \end{pmatrix}^T \tag{3.14}$$

However, the use of a projection matrix to derive $\overrightarrow{\mathbf{v}}_j$ is inefficient as the results of many calculations are discarded at the final step. It is much more computationally efficient to carry out all arithmetic using our original $1 \times N$ vectors $\overrightarrow{\mathbf{v}}_j$ instead of the much larger $1 \times NM$ vectors $\overrightarrow{\mathbf{w}}_j$.

3.5.6 Shape of the eigenvectors of A

Proposition 3. *The eigenvectors $\overrightarrow{\boldsymbol{\psi}}_{k,1}$ with corresponding eigenvalue λ_k of the matrix $\underline{\mathbf{A}}$ (3.7) are of the form*

$$\overrightarrow{\boldsymbol{\psi}}_{k,1} = \begin{pmatrix} \overrightarrow{\mathbf{a}} & \lambda_k \overrightarrow{\mathbf{a}} & \cdots & \lambda_k^{M-1} \overrightarrow{\mathbf{a}} \end{pmatrix}$$

where $\overrightarrow{\mathbf{a}}$ is a vector in $\mathbb{R}^N$.

Proof

Let $\overrightarrow{\boldsymbol{\psi}}_{k,1} = (\overrightarrow{\mathbf{a}}_0, \overrightarrow{\mathbf{a}}_1, \ldots, \overrightarrow{\mathbf{a}}_{M-1})$ be an eigenvector of $\underline{\mathbf{A}}$ given by (3.7) with eigenvalue λ_k, where the $\overrightarrow{\mathbf{a}}_i$ are simply the appropriate components of $\overrightarrow{\boldsymbol{\psi}}_{k,1}$. From the definition of an eigenvector, we have

$$\overrightarrow{\boldsymbol{\psi}}_{k,1} \underline{\mathbf{A}} = \lambda_k \overrightarrow{\boldsymbol{\psi}}_{k,1}$$

We can now separate this into individual equations involving the component vectors $\overrightarrow{\mathbf{a}}_i$, so that

$$\begin{gathered} \overrightarrow{\mathbf{a}}_1 = \lambda_k \overrightarrow{\mathbf{a}}_0 \\ \overrightarrow{\mathbf{a}}_2 = \lambda_k \overrightarrow{\mathbf{a}}_1 \\ \vdots \\ \overrightarrow{\mathbf{a}}_{M-1} = \lambda_k \overrightarrow{\mathbf{a}}_{M-2} \\ -\sum_{k=0}^{M-1} \overrightarrow{\mathbf{a}}_k \underline{\mathbf{Q}}_k \underline{\mathbf{Q}}_M^{-1} = \lambda_k \overrightarrow{\mathbf{a}}_{M-1} \end{gathered}$$

From the first $M-1$ equations it follows that $\overrightarrow{\boldsymbol{\psi}}_{k,1}=(\overrightarrow{\mathbf{a}}_0, \lambda_k \overrightarrow{\mathbf{a}}_0, \lambda_k^2 \overrightarrow{\mathbf{a}}_0, \ldots, \lambda_k^{M-1} \overrightarrow{\mathbf{a}}_0)$. By setting $\overrightarrow{\mathbf{a}} = \overrightarrow{\mathbf{a}}_0$ we get the desired result. ■

3.5.7 Shape of the first generalised eigenvector of A

We will now derive the general shape of a generalised eigenvector of matrix $\underline{\mathbf{A}}$. At first, we will be considering the first generalised eigenvector of the k^{th} Jordan block, *i.e.* $\overrightarrow{\boldsymbol{\psi}}_{k,2}$. Later we will consider the general form of $\overrightarrow{\boldsymbol{\psi}}_{k,i}$. We know that the eigenvector for this Jordan block satisfies

$$\overrightarrow{\boldsymbol{\psi}}_{k,1} = (\overrightarrow{\mathbf{a}}_0, \lambda_k \overrightarrow{\mathbf{a}}_0, \lambda_k^2 \overrightarrow{\mathbf{a}}_0, \ldots, \lambda_k^{M-1} \overrightarrow{\mathbf{a}}_0)$$

Let $\overrightarrow{\boldsymbol{\psi}}_{k,2} = (\overrightarrow{\mathbf{b}}_0, \overrightarrow{\mathbf{b}}_1, \ldots, \overrightarrow{\mathbf{b}}_{M-1})$ be a generalised eigenvector of $\underline{\mathbf{A}}$ we chose by using the recurrence relation

$$\overrightarrow{\boldsymbol{\psi}}_{k,2}(\underline{\mathbf{A}} - \lambda_k \underline{\mathbf{I}}) = \overrightarrow{\boldsymbol{\psi}}_{k,1}$$

Here the $\overrightarrow{\mathbf{b}}_i$ are simply the appropriate components of $\overrightarrow{\boldsymbol{\psi}}_{k,2}$. By re-arranging the recurrence relation, we get

$$\overrightarrow{\boldsymbol{\psi}}_{k,2}\underline{\mathbf{A}} = \lambda_k \overrightarrow{\boldsymbol{\psi}}_{k,2} + \overrightarrow{\boldsymbol{\psi}}_{k,1}$$

We now separate this equation into individual ones involving the component vectors $\overrightarrow{\mathbf{a}}_i$ and $\overrightarrow{\mathbf{b}}_i$, so that

$$\begin{aligned}
\overrightarrow{\mathbf{b}}_1 &= \lambda_k \overrightarrow{\mathbf{b}}_0 + \overrightarrow{\mathbf{a}}_0 \\
\overrightarrow{\mathbf{b}}_2 &= \lambda_k \overrightarrow{\mathbf{b}}_1 + \overrightarrow{\mathbf{a}}_1 \\
&\vdots \\
\overrightarrow{\mathbf{b}}_{M-1} &= \lambda_k \overrightarrow{\mathbf{b}}_{M-2} + \overrightarrow{\mathbf{a}}_{M-2} \\
-\textstyle\sum_{k=0}^{M-1} \overrightarrow{\mathbf{b}}_k \underline{\mathbf{Q}}_k \underline{\mathbf{Q}}_M^{-1} &= \lambda_k \overrightarrow{\mathbf{b}}_{M-1} + \overrightarrow{\mathbf{a}}_{M-1}
\end{aligned}$$

It follows that

$$\overrightarrow{\boldsymbol{\psi}}_{k,2} = (\overrightarrow{\mathbf{b}}_0, \lambda_k \overrightarrow{\mathbf{b}}_0 + \overrightarrow{\mathbf{a}}_0, \lambda_k \overrightarrow{\mathbf{b}}_1 + \overrightarrow{\mathbf{a}}_1, \ldots, \lambda_k \overrightarrow{\mathbf{b}}_{M-2} + \overrightarrow{\mathbf{a}}_{M-2})$$

Remembering that the $\overrightarrow{\mathbf{a}}_i$ are the component vectors of the eigenvector and hence satisfy $\overrightarrow{\mathbf{a}}_i = \lambda_k^i \overrightarrow{\mathbf{a}}_0$, we can re-write the third component vector of $\overrightarrow{\boldsymbol{\psi}}_{k,2}$ as follows

$$\lambda_k \overrightarrow{\mathbf{b}}_1 + \overrightarrow{\mathbf{a}}_1 = \lambda_k(\lambda_k \overrightarrow{\mathbf{b}}_0 + \overrightarrow{\mathbf{a}}_0) + \lambda_k \overrightarrow{\mathbf{a}}_0 = \lambda_k^2 \overrightarrow{\mathbf{b}}_0 + 2\lambda_k \overrightarrow{\mathbf{a}}_0$$

using a similar procedure on the other component vectors of $\overrightarrow{\boldsymbol{\psi}}_{k,2}$, we find

$$\overrightarrow{\boldsymbol{\psi}}_{k,2} = (\overrightarrow{\mathbf{b}}_0, \lambda_k \overrightarrow{\mathbf{b}}_0 + \overrightarrow{\mathbf{a}}_0, \lambda_k^2 \overrightarrow{\mathbf{b}}_0 + 2\lambda_k \overrightarrow{\mathbf{a}}_0, \ldots, \lambda_k^{M-1} \overrightarrow{\mathbf{b}}_0 + (M-1)\lambda_k^{M-2} \overrightarrow{\mathbf{a}}_0)$$

or,

$$\begin{aligned} \overrightarrow{\boldsymbol{\psi}}_{k,2} &= (\overrightarrow{\mathbf{b}}_0, \lambda_k \overrightarrow{\mathbf{b}}_0, \lambda_k^2 \overrightarrow{\mathbf{b}}_0, \ldots, \lambda_k^{M-1} \overrightarrow{\mathbf{b}}_0) \\ &+ (\overrightarrow{\mathbf{0}}, \overrightarrow{\mathbf{a}}_0, 2\lambda_k \overrightarrow{\mathbf{a}}_0, \ldots, (M-1)\lambda_k^{M-2} \overrightarrow{\mathbf{a}}_0) \end{aligned}$$

3.5.8 Shape of the general generalised eigenvector

We now derive the shape of the i^{th} generalised eigenvector of the k^{th} Jordan block, $\overrightarrow{\boldsymbol{\psi}}_{k,i}$ in relation to its component vectors. Let

$$\overrightarrow{\boldsymbol{\psi}}_{k,i} = (\overrightarrow{\mathbf{c}}_{i,0}, \overrightarrow{\mathbf{c}}_{i,1}, \ldots, \overrightarrow{\mathbf{c}}_{i,M-1}) \tag{3.15}$$

where $\overrightarrow{\mathbf{c}}_{i,l}$ are the appropriate component vectors, just as during the derivation of the shape of the eigenvector and the first generalised eigenvector above. Note that for clarity we do not add the k-subscript to the $\overrightarrow{\mathbf{c}}_{i,l}$, as it is understood that we are operating within the k^{th} Jordan block throughout the calculations. Also, using the notation from sections 3.5.6 and 3.5.7 we see that $\overrightarrow{\mathbf{c}}_{1,l} = \overrightarrow{\mathbf{a}}_l$ and $\overrightarrow{\mathbf{c}}_{2,l} = \overrightarrow{\mathbf{b}}_l$.

We now transform (3.15) so that it only involves component vectors of the form $\overrightarrow{\mathbf{c}}_{y,0}$ by using the following replacement rules

$$\begin{aligned} \overrightarrow{\mathbf{c}}_{1,x} &= \lambda_k \overrightarrow{\mathbf{c}}_{1,x-1} && x \geq 1 && \textit{c.f.} \text{ section 3.5.6} \\ \overrightarrow{\mathbf{c}}_{y,x} &= \lambda_k \overrightarrow{\mathbf{c}}_{y,x-1} + \overrightarrow{\mathbf{c}}_{y-1,x-1} && y \geq 2, x \geq 1 && \textit{c.f.} \text{ section 3.5.7} \end{aligned} \tag{3.16}$$

Proposition 4. *Recursively using (3.16) on (3.15) gives*

$$\overrightarrow{\boldsymbol{\psi}}_{k,i} = \begin{pmatrix} \overrightarrow{\mathbf{c}}_{i,0} \\ \lambda_k \overrightarrow{\mathbf{c}}_{i,0} + \overrightarrow{\mathbf{c}}_{i-1,0} \\ \begin{pmatrix} 2 \\ 0 \end{pmatrix} \lambda_k^0 \overrightarrow{\mathbf{c}}_{i,0} + \begin{pmatrix} 2 \\ 1 \end{pmatrix} \lambda_k \overrightarrow{\mathbf{c}}_{i-1,0} + \begin{pmatrix} 2 \\ 2 \end{pmatrix} \lambda_k^2 \overrightarrow{\mathbf{c}}_{i-2,0} \\ \vdots \\ \begin{pmatrix} M-1 \\ 0 \end{pmatrix} \lambda_k^0 \overrightarrow{\mathbf{c}}_{i,0} + \cdots + \begin{pmatrix} M-1 \\ M-1 \end{pmatrix} \lambda_k^{M-1} \overrightarrow{\mathbf{c}}_{i-(M-1),0} \end{pmatrix}^T$$

Or equivalently as the sum of M vectors, each involving only one of the component vectors,

$$\begin{aligned}
\overrightarrow{\boldsymbol{\psi}}_{k,i} &= \left(\overrightarrow{\mathbf{c}}_{i,0}, \lambda_k \overrightarrow{\mathbf{c}}_{i,0}, \begin{pmatrix} 2 \\ 0 \end{pmatrix} \lambda_k^2 \overrightarrow{\mathbf{c}}_{i,0}, \ldots \begin{pmatrix} M-1 \\ 0 \end{pmatrix} \lambda_k^{M-1} \overrightarrow{\mathbf{c}}_{i,0} \right) \\
&+ \left(\overrightarrow{\mathbf{0}}, \overrightarrow{\mathbf{c}}_{i-1,0}, \begin{pmatrix} 2 \\ 1 \end{pmatrix} \lambda_k \overrightarrow{\mathbf{c}}_{i-1,0} + \cdots + \begin{pmatrix} M-1 \\ 1 \end{pmatrix} \lambda_k^{M-2} \overrightarrow{\mathbf{c}}_{i-1,0} \right) \\
&+ \vdots \\
&+ \left(\overrightarrow{\mathbf{0}}, \overrightarrow{\mathbf{0}}, \ldots, \overrightarrow{\mathbf{0}}, \begin{pmatrix} M-1 \\ M-1 \end{pmatrix} \lambda_k^0 \overrightarrow{\mathbf{c}}_{i-(M-1),0} \right)
\end{aligned}$$

Proof

We need to show that the j^{th} component vector of $\overrightarrow{\boldsymbol{\psi}}_{k,i}$ (*i.e.* $\overrightarrow{\mathbf{c}}_{i,j}$) when written in terms of $\overrightarrow{\mathbf{c}}_{y,0}$, is given by

$$\overrightarrow{\mathbf{c}}_{i,j} = \sum_{l=0}^{j} \begin{pmatrix} j \\ l \end{pmatrix} \lambda_k^l \overrightarrow{\mathbf{c}}_{i-l,0}$$

where $\overrightarrow{\mathbf{c}}_{k,0} = \overrightarrow{\mathbf{0}}$ whenever $k \leq 0$.

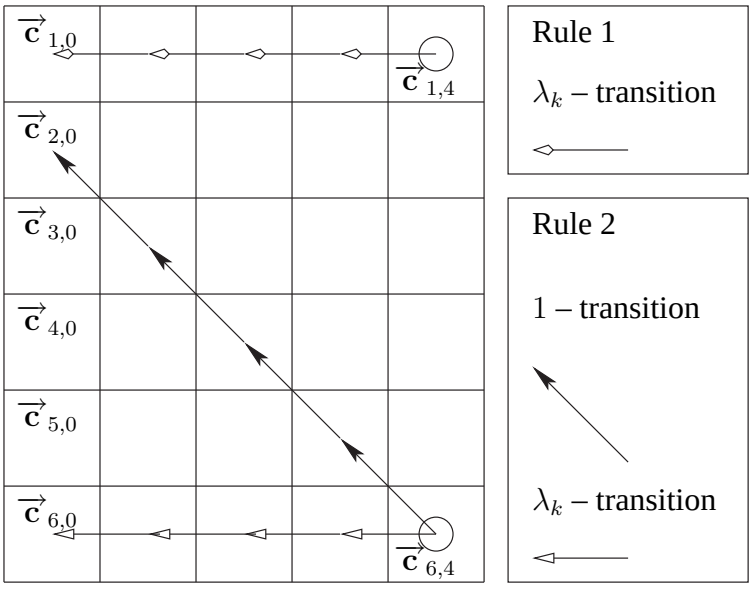


Figure 3.4: Single paths using either rule

To show this, we notice that the replacement rules (3.16) can be mapped to two types of transitions on a two-dimensional grid, on which coordinates are given by the subscripts

i and j of $\overrightarrow{\mathbf{c}}_{i,j}$. The first rule in (3.16) is only applicable when $i = 1$ and represents a transition to the left of the grid. It is labelled λ_k-transition, because it specifies a multiplication by λ_k. The second rule is valid for $i > 1$ and specifies two transitions, one going up and left with no change of quantity (multiplication factor 1) and the other going left, again with multiplication factor λ_k.

Figure 3.4 depicts the procedure we apply to represent $\overrightarrow{\mathbf{c}}_{6,4}$ and $\overrightarrow{\mathbf{c}}_{1,4}$ in terms of a selection of $\overrightarrow{\mathbf{c}}_{y,0}$ (the leftmost column) that are simple to find. They are simple to find due to the fact that there is a unique path from the starting point to the end points. The diagram tells us the following

$$\overrightarrow{\mathbf{c}}_{1,4} = \lambda_k^4 \overrightarrow{\mathbf{c}}_{1,0}$$

$$\text{The } \overrightarrow{\mathbf{c}}_{2,0} \text{ component of } \overrightarrow{\mathbf{c}}_{6,4} \text{ is } 1\overrightarrow{\mathbf{c}}_{2,0}$$

$$\text{The } \overrightarrow{\mathbf{c}}_{6,0} \text{ component of } \overrightarrow{\mathbf{c}}_{6,4} \text{ is } \lambda_k^4 \overrightarrow{\mathbf{c}}_{6,0}$$

When multiple paths exist, we need to sum up the contributions from each one. Figure 3.5 illustrates this using an example calculating the $\overrightarrow{\mathbf{c}}_{3,0}$-component of $\overrightarrow{\mathbf{c}}_{6,4}$. There are a total of 4 paths that lead from $(6, 4)$ to $(3, 0)$, and along the way, the multipliers are $1^3\lambda_k = \lambda_k$, because exactly one horizontal transition must be used, hence the component of $\overrightarrow{\mathbf{c}}_{3,0}$ of $\overrightarrow{\mathbf{c}}_{6,4}$ is $4\lambda_k$. In the general case, there are $\begin{pmatrix} j \\ l \end{pmatrix}$ paths from $\overrightarrow{\mathbf{c}}_{i,j}$ to $\overrightarrow{\mathbf{c}}_{i-l,0}$, each of which involves the cumulative multiplier λ_k^l. Hence, when $\overrightarrow{\mathbf{c}}_{i,j}$ is re-written in terms of $\overrightarrow{\mathbf{c}}_{x,0}$, we get

$$\overrightarrow{\mathbf{c}}_{i,j} = \sum_{l=0}^{j} \begin{pmatrix} j \\ l \end{pmatrix} \lambda_k^l \overrightarrow{\mathbf{c}}_{i-l,0}$$

In the above case rule 2 is exclusively used, because we were considering $\overrightarrow{\mathbf{c}}_{i,j}$, where $i > j$. When $i \leq j$, a mixture of both rules has to be used for the $\overrightarrow{\mathbf{c}}_{1,0}$ component. This situation is depicted in figure 3.6. Note the distinct types of arrows for the horizontal transition.

Due to the fact that both rules agree in the multiplier used for the horizontal transitions, we obtain the same result for the $\overrightarrow{\mathbf{c}}_{1,0}$ component as if only rule 2 had been used in reaching $(1, 0)$, even-though it does not apply within the first row. The sum expressing

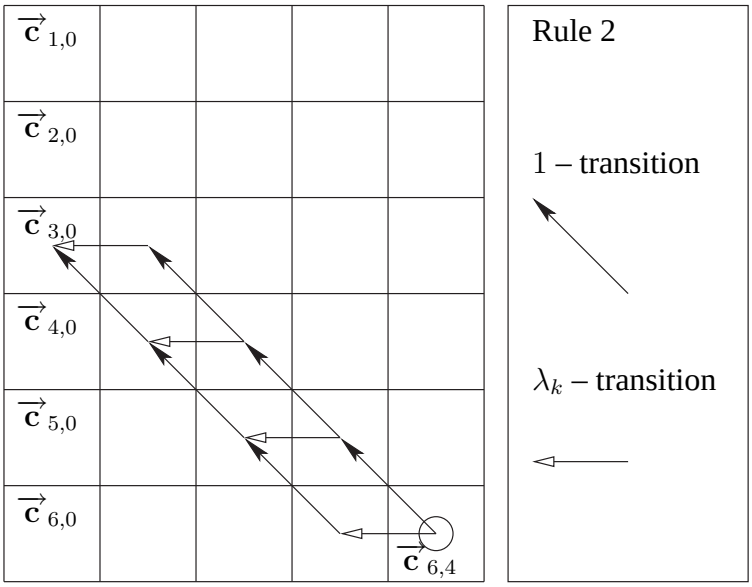


Figure 3.5: Multiple Paths: evaluating the $c_{y,0}$ component of $c_{i,j}$ when $i > j$

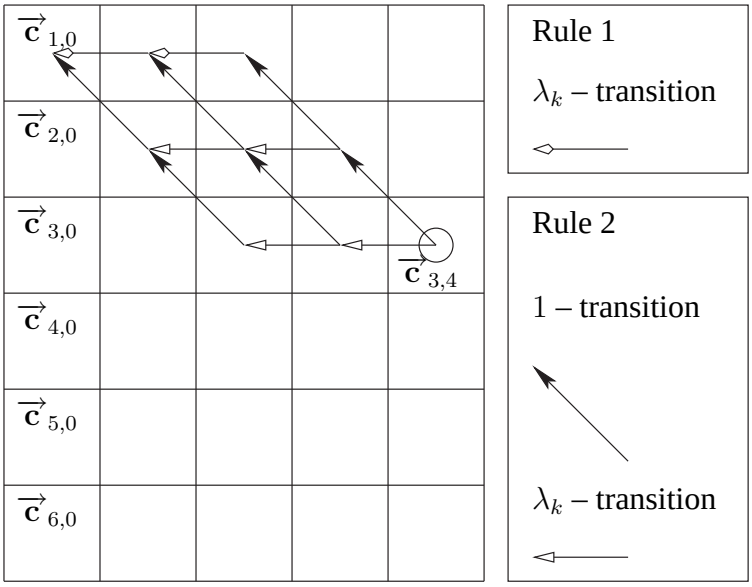


Figure 3.6: evaluating the $c_{1,0}$ component of $c_{i,j}$ when $i \leq j$

$\overrightarrow{\mathbf{c}}_{i,j}$ has to cover a smaller range when $i \leq j$ as there are no terms $\overrightarrow{\mathbf{c}}_{y,0}$ for $y \leq 0$. Consequently, we have the following.

$$\overrightarrow{\mathbf{c}}_{i,j} = \sum_{l=0}^{j} \begin{pmatrix} j \\ l \end{pmatrix} \lambda_k^l \overrightarrow{\mathbf{c}}_{i-l,0} \quad i > j$$

$$\overrightarrow{\mathbf{c}}_{i,j} = \sum_{l=0}^{i-1} \begin{pmatrix} j \\ l \end{pmatrix} \lambda_k^l \overrightarrow{\mathbf{c}}_{i-l,0} \quad i \leq j$$

To unify the two expressions, we define $\overrightarrow{\mathbf{c}}_{y,0} = \overrightarrow{\mathbf{0}}$ whenever $y \leq 0$, giving

$$\overrightarrow{\mathbf{c}}_{i,j} = \sum_{l=0}^{j} \begin{pmatrix} j \\ l \end{pmatrix} \lambda_k^l \overrightarrow{\mathbf{c}}_{i-l,0} \quad \forall i \geq 1, j \geq 0$$

which is the required expression. ■

We can now express the solution without the need for a projection matrix as follows:

$$\overrightarrow{\mathbf{v}}_{\epsilon^b+j} = \sum_{k=1}^{r} \left[\sum_{l=1}^{s_k} \alpha_{k,l} \left[\sum_{m=0}^{l-1} \begin{pmatrix} j \\ m \end{pmatrix} \lambda_k^{j-m} \overrightarrow{\boldsymbol{\zeta}}_{k,l-m} \right] \right]$$

Again, if all Jordan blocks $1 \ldots r$ of $\underline{\mathbf{A}}$ are of size $s_k = 1$, we obtain the familiar spectral expansion representation

$$\overrightarrow{\mathbf{v}}_{\epsilon^b+j} = \sum_{k=1}^{NM} \alpha_{k,1} \lambda_k^j \overrightarrow{\boldsymbol{\zeta}}_{k,1}$$

In practice the need for generalised eigenvectors in addition to simple eigenvectors has, so far, only been found to arise in a few special cases. As mentioned previously, one of these is given by a finite queue at the exact saturation point, *i.e.* where the mean customer arrival rate exactly equals the mean customer departure rate (due to service completions or negative customer arrivals). In this case, a generalised eigenvector with eigenvalue 1 exists. This generalised eigenvector, conjointly with the eigenvector $\overrightarrow{\pi}$ given by the steady-state of the modulator $\overrightarrow{\pi} Q = \overrightarrow{\mathbf{0}}$, that is present in the solution to every queue, form a Jordan block of size 2. Another possibility has already been mentioned in a different context in section 3.4.1, when discussing complex eigenvalues. Changing the *speed* of modulation by multiplying the modulation matrix $\underline{\mathbf{Q}}$ by an

appropriate scalar, k, can result in a situation where two eigenvalues coincide. In the particular example the value found was at approximatively $k = 0.27$. Unfortunately the queue cannot be examined at the exact point where the double eigenvalue occurs, as the solution for k is not symbolically possible.

3.6 Matrix Geometric solution methods

Unlike spectral expansion, which represents the probabilities within the repeating region ($\epsilon^b \leq j \leq \epsilon^t$) using eigenvalues and vectors, matrix geometric methods [Neu81] utilise a non-singular matrix $\underline{\mathbf{F}}$ to describe changes in probability between two adjacent levels *i.e.*

$$\overrightarrow{\mathbf{v}}_{j+1} = \overrightarrow{\mathbf{v}}_j \underline{\mathbf{F}} \Leftrightarrow \overrightarrow{\mathbf{v}}_{j+i} = \overrightarrow{\mathbf{v}}_j \underline{\mathbf{F}}^i \qquad \epsilon^b \leq j \leq j+i \leq \epsilon^t \tag{3.17}$$

or, by anchoring the expression at the bottom of the repeating region,

$$\overrightarrow{\mathbf{v}}_j = \overrightarrow{\mathbf{f}} \underline{\mathbf{F}}^{j-\epsilon^b} \qquad \epsilon^b \leq j \leq \epsilon^t \tag{3.18}$$

Here, $\overrightarrow{\mathbf{f}}$ takes the place of the α's in SE to be constrained by the boundary conditions imposed by the rest of the queue *i.e.* the processing region, and in the case of a finite queue, the blocking region. The power we raise the matrix $\underline{\mathbf{F}}$ to could be offset by any integer as any change here can be absorbed in a corresponding change in the value of $\overrightarrow{\mathbf{f}}$. We have chosen to use $\underline{\mathbf{F}}^{j-\epsilon^b}$ because this both minimises the powers that $\underline{\mathbf{F}}$ has to be raised to in calculations and ensures that $\overrightarrow{\mathbf{v}}_{\epsilon^b} = \overrightarrow{\mathbf{f}}$

By substituting (3.18) into the matrix recurrence in section 3.1 we see that a solution using this representation can only exist if $\underline{\mathbf{F}}$ satisfies the matrix polynomial equation

$$\sum_{k=0}^{M} \underline{\mathbf{F}}^k \underline{\mathbf{Q}}_k = \underline{\mathbf{0}} \tag{3.19}$$

Efficient methods of solving for $\underline{\mathbf{F}}$ are presented in [LatRam93] and [NaoKri+97].

3.6.1 Correspondence in eigenmodes

Let $\underline{\hat{\mathbf{F}}}$ be a non-singular matrix found using one of the above methods that satisfies the matrix polynomial (3.19). Having full rank N, the matrix $\underline{\hat{\mathbf{F}}}$ has N non-zero

eigenvalues ζ_n and corresponding eigenvectors $\overrightarrow{\boldsymbol{\chi}}_n$.

We can uniquely express the vector $\overrightarrow{\mathbf{v}}_{\epsilon^b}$ as a linear combination of these eigenvectors,

$$\overrightarrow{\mathbf{v}}_{\epsilon^b} = \sum_{n=1}^{N} \gamma_n \overrightarrow{\boldsymbol{\chi}}_n \tag{3.20}$$

Then from equation (3.18) we have

$$\overrightarrow{\mathbf{v}}_j = \sum_{n=1}^{N} \gamma_n \zeta_n^{j-\epsilon^b} \overrightarrow{\boldsymbol{\chi}}_n \qquad \epsilon^b \leq j \leq \epsilon^t \tag{3.21}$$

We now compare this expression to the one derived for spectral expansion

$$\overrightarrow{\mathbf{v}}_j = \sum_{i=1}^{n^*} \alpha_i \xi_i^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t \tag{3.22}$$

If the Matrix Geometric method gives a correct solution, it must be equal to that derived when using generalised spectral expansion, *i.e.*

$$\sum_{n=1}^{N} \gamma_n \zeta_n^{j-\epsilon^b} \overrightarrow{\boldsymbol{\chi}}_n = \sum_{i=1}^{n^*} \alpha_i \xi_i^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t \tag{3.23}$$

Proposition 5. *Any non-zero eigenvalue ζ_n of $\underline{\hat{\mathbf{F}}}$ (found using matrix geometric methods) is equal to an eigenvalue ξ_m found during generalised spectral expansion and the sets of non-zero eigenvectors $\zeta = \{\zeta_n\}$ and $\xi = \{\xi_m\}$ satisfy $\zeta \subseteq \xi$.*

Proof

Assume that ζ is not a subset of ξ. Therefore there exists a non-zero eigenvalue $\zeta_m \in \zeta$ such that $\zeta_m \notin \xi$. The equality (3.23) holds for any choice of the free variables γ_n, so that we chose $\gamma_n = \delta_{n,m}$ and try to determine the values of the free variables, α_i that the generalised spectral expansion method uses to represent this particular solution.

$$\sum_{n=1}^{N} \delta_{n,m} \zeta_n^{j-\epsilon^b} \overrightarrow{\boldsymbol{\chi}}_n = \sum_{i=1}^{n^*} \alpha_i \xi_i^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t \tag{3.24}$$

$$\zeta_m^{j-\epsilon^b} \overrightarrow{\boldsymbol{\chi}}_m = \sum_{i=1}^{n^*} \alpha_i \xi_i^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t$$

$$\overrightarrow{\boldsymbol{\chi}}_m = \sum_{i=1}^{n^*} \alpha_i \frac{\xi_i^j}{\zeta_m^{j-\epsilon^b}} \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t$$

$$\overrightarrow{\boldsymbol{\chi}}_m = \sum_{i=1}^{n^*} \frac{\alpha_i}{\zeta_m^{\epsilon^b}} \left(\frac{\xi_i}{\zeta_m}\right)^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t$$

The LHS of this expression is constant, whereas the right hand side varies with j, except in the following two cases.

Case 1. $\alpha_i = 0$ for all i. From this it follows that an eigenvector $\overrightarrow{\boldsymbol{\chi}}_m = \overrightarrow{\mathbf{0}}$. This contradicts the assumption that the matrix $\underline{\mathbf{F}}$ found using matrix geometric methods has full rank.

Case 2. Either $\frac{\xi_i}{\zeta_m} = 1$ or $\alpha_i = 0$ for all $1 \leq i \leq n^*$. It follows that for at least one i, we have $\xi_i = \zeta_m$. This contradicts the assumption that $\zeta_m \notin \xi$. ■

Proposition 6. *Let $\overrightarrow{\boldsymbol{\chi}}$ be the set of non-zero eigenvectors of the matrix geometric solution matrix $\underline{\mathbf{F}}$, and similarly $\overrightarrow{\boldsymbol{\psi}}$ the set of eigenvectors found using spectral expansion. The set of (non-zero) eigen*vectors *satisfy the property that any member $\overrightarrow{\boldsymbol{\chi}}_m \in \overrightarrow{\boldsymbol{\chi}}$ lies within a subspace*[6] *spanned by members $\overrightarrow{\boldsymbol{\psi}}_n \in \overrightarrow{\boldsymbol{\psi}}$, with the same associated eigenvalue.*

Proof

Again, we chose $\gamma_n = \delta_{n,m}$ in the expression (3.23) and hence equation (3.24).

Since there is no j-dependence on the RHS, $\alpha_i = 0$ whenever $\xi_i \neq \zeta_m$. From the above, we know that there is at least one i such that $\xi_i = \zeta_m$. Let there be a total of $k \geq 1$ equal eigenvalues. After re-labelling all terms for which we did not set the coefficient $\alpha_i = 0$, we have

$$\overrightarrow{\boldsymbol{\chi}}_m = \sum_{i=1}^{k} \frac{\alpha_i}{\zeta_m^{\epsilon^b}} \left(\frac{\xi_i}{\zeta_m}\right)^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t$$

$$\textit{i.e.} \quad \overrightarrow{\boldsymbol{\chi}}_m = \sum_{i=1}^{k} \frac{\alpha_i}{\zeta_m^{\epsilon^b}} 1^j \overrightarrow{\boldsymbol{\psi}}_i \qquad \epsilon^b \leq j \leq \epsilon^t$$

It follows that $\overrightarrow{\boldsymbol{\chi}}_m$ is a vector that lies within the subspace spanned by $\left\{\overrightarrow{\boldsymbol{\psi}}_i\right\}_{1 \leq i \leq k}$, where the spanning vectors are all eigenvectors of the SE method with the same eigenvalue, ζ_m. ■

[6] Usually, this subspace only has dimension 1, and hence the eigenvectors are pairwise parallel

3.6.2 Backward series

So far we have identified a subset of size N of the eigenvalues and eigenvectors contained within the complete solution. To gain access to additional eigenmodes, practitioners of the matrix geometric methods use the backward series

$$\sum_{k=0}^{M} \overrightarrow{\mathbf{v}}_{j+k} \underline{\mathbf{Q}}_{M-k} = \underline{\mathbf{0}} \tag{3.25}$$

This matrix recurrence has the same structure as the original forward series given in (3.5) involving $\underline{\mathbf{F}}$ and we can use the same algorithms to find a numerical approximation $\underline{\hat{\mathbf{B}}}$ to $\underline{\mathbf{B}}$ with $\overrightarrow{\mathbf{v}}_{j+1} = \overrightarrow{\mathbf{v}}_j \underline{\mathbf{B}}$, a solution to

$$\sum_{k=0}^{M} \underline{\mathbf{B}}^k \underline{\mathbf{Q}}_{M-k} = \underline{\mathbf{0}} \tag{3.26}$$

When using the forward and backward series, the repeating region is represented as follows

$$\overrightarrow{\mathbf{v}}_j = \overrightarrow{\mathbf{v}}_{\epsilon^b} \underline{\mathbf{F}}^{j-\epsilon^b} + \overrightarrow{\mathbf{v}}_{\epsilon^t} \underline{\mathbf{B}}^{\epsilon^t - j}$$

This representation is only well-defined when the queue is finite. For infinite queues the backward series cannot be used.

For an eigenvalue λ and eigenvector $\overrightarrow{\psi}$ of $\underline{\mathbf{B}}$

$$\overrightarrow{\psi}\underline{\mathbf{B}} = \lambda \overrightarrow{\psi} \Leftrightarrow \overrightarrow{\psi}\underline{\mathbf{B}}^{-1} = \frac{1}{\lambda}\overrightarrow{\psi}$$

We note that the inverse of the backward matrix, $\underline{\hat{\mathbf{B}}}^{-1}$, shares exactly N eigenvalues and left eigenvectors with the solution set provided by generalised spectral expansion, where the eigenvalues are the reciprocals of those within $\underline{\mathbf{B}}$. Hence, taken collectively, the matrices $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}$ provides exactly $2N$ distinct eigenvalues and eigenvectors (a total of N for an infinite queue).

Depending on the number of eigenvalues $n^* = |\xi|$ found by general spectral expansion in relation to those of the matrix geometric method, which is necessarily fixed at $2N$, we can make three observations.

• If $n^* < 2N$, there are fewer eigenmodes present in the recurrence relation than both matrices contain. There are $2N - n^*$ eigenvalues between $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}^{-1}$ that are not constrained. An arbitrary choice of eigenvalues and eigenvectors for these matrices will

change the solution, unless we either use zero-eigenvalues (paired with some linearly independent eigenvectors) or use a number of eigenvector/eigenvalue pairs twice, once for $\underline{\hat{\mathbf{F}}}$ and another time for $\underline{\hat{\mathbf{B}}}^{-1}$. The introduction of zero-eigenmodes will make one of $\underline{\mathbf{F}}$ and $\underline{\mathbf{B}}$ singular, whereas using the same eigenmode twice yields two non-singular matrices. While normally singular matrices are to be avoided, it turns out that in practice the former method is numerically more stable for the subsequent solution process in which $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}^{-1}$ will be used. When using the formula $\overrightarrow{\mathbf{v}}_j = \overrightarrow{\mathbf{v}}_{\epsilon^b}\underline{\mathbf{F}}^{j-\epsilon^b} + \overrightarrow{\mathbf{v}}_{\epsilon^t}\underline{\mathbf{B}}^{\epsilon^t-j}$ to calculate, we raise both matrices to possibly large powers. This process is unproblematic for a singular matrix because its zero-subspace maps vectors to zero, thereby not allowing any errors to grow when raised to a power.

• When $n^* = 2N$, we have a direct correspondence in the solutions given by the spectral expansion and matrix geometric methods, with the only difference lying in the representation of the solution using either (3.4) or (3.18). We can seamlessly move from the matrix geometric method to spectral expansion by finding the eigenvalues of $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}$, and back by constructing $\underline{\hat{\mathbf{F}}} = \underline{\mathbf{\Psi}}^{-1}\underline{\mathbf{\Xi}}\,\underline{\mathbf{\Psi}}$ (similarly for $\underline{\hat{\mathbf{B}}}$), where the i-th column of $\underline{\mathbf{\Psi}}$ is $\overrightarrow{\boldsymbol{\psi}}_i$ and $\underline{\mathbf{\Xi}}$ is a diagonal matrix with i-th element ξ_i. Then, *e.g.*

$$\overrightarrow{\boldsymbol{\psi}}_i\underline{\hat{\mathbf{F}}} = \overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{\Psi}}^{-1}\underline{\mathbf{\Xi}}\,\underline{\mathbf{\Psi}} = \overrightarrow{\mathbf{e}}_i\underline{\mathbf{\Xi}}\,\underline{\mathbf{\Psi}} = \xi_i\overrightarrow{\mathbf{e}}_i\underline{\mathbf{\Psi}} = \xi_i\overrightarrow{\boldsymbol{\psi}}_i$$

• In the case $n^* > 2N$, there are more eigenmodes present in the recurrence relation than can be represented within two matrices $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}$ of rank N. In terms of the spectral expansion representation of a solution derived using the matrix geometric method, this forces the coefficients α_i in (3.4) to be zero whenever an eigenmode i is not represented by either the forward of backward matrix. Depending on the parameters of the queue model being solved, this can invalidate a solution obtained through the matrix geometric solution.

3.6.3 Modified MG method using multiple matrices R

Should just one pair of matrices $(\underline{\hat{\mathbf{F}}}, \underline{\hat{\mathbf{B}}})$ be insufficient in accurately representing all present eigenmodes in the system, multiple solution matrices $\underline{\mathbf{F}}_k$ $(1 \leq k \leq k^F)$ and $\underline{\mathbf{B}}_k$ $(1 \leq k \leq k^B)$ of (3.19) and (3.26) have to be sought. The augmented probability

representation is

$$\overrightarrow{\mathbf{v}}_j = \sum_{k=1}^{k^F} \overrightarrow{\mathbf{v}}_{k,\epsilon^b} \underline{\mathbf{F}}_k^{j-\epsilon^b} + \sum_{k=1}^{k^B} \overrightarrow{\mathbf{v}}_{k,\epsilon^t} \underline{\mathbf{B}}_k^{\epsilon^t - j} \tag{3.27}$$

The introduction of multiple solution matrices necessitates the presence of additional boundary conditions given by $\overrightarrow{\mathbf{v}}_{k,\epsilon^b}$ and $\overrightarrow{\mathbf{v}}_{k,\epsilon^t}$. This is only to be expected as the Spectral Expansion method also allocated one boundary condition per stream and the Matrix Geometric method is simply an alternative representation of a solution within a repeating region.

As before it is the case that the eigenvalues and eigenvectors of matrix geometric solution matrices $\underline{\mathbf{F}}_k$ and $\underline{\mathbf{B}}_k^{-1}$ are subsets of ξ and $\overrightarrow{\psi}$ respectively. To minimise the number $k^F + k^B$ of necessary matrices and with it the computational complexity of the modified matrix geometric representation, care should be taken to choose solution matrices such that the sets of their eigenvalues (and eigenvectors) are pairwise disjoint.

A potential difficulty arises from the fixed size N of all matrices involved. If the number of necessary eigenmodes n^* is not divisible by N, one of the solution-matrices will have insufficient eigenvalues (and eigenvectors) to be fully specified. To remedy this situation, we pad the representation with linearly independent zero-eigenvalue eigenvectors.

So far no algorithm is known that does not involve prior knowledge of all eigenvalues and eigenvectors. Later we present a transformation that can be applied to the state space of a queue which allows the solution of $n^* > 2N$ problems using matrix geometric methods.

3.6.4 Limitations of the MG method

Our experimentation with spectral expansion has revealed important characteristics of a class of queues which cannot be solved using matrix geometric methods in their current form. The problem lies in the spectrum of the solution. Matrices as used in MG have a fixed number of eigenvalues, equal to the number N of modulation states. Currently known algorithms provide either one or two of these matrices, depending on whether the queue is infinite of finite.

Infinite Queues

Infinite queues do not require the full set of eigenvalues, in particular only a number n^* of those which have magnitude strictly less than one. The value of n^* has been found to be directly related to the number of distinct arrival processes, n^{arr}, over all modulation states and for an infinite queue that is not saturated. In practice we find, disregarding degenerate[7] arrival streams, $n^* = n^{arr}N$.

The MG method only provides one matrix, $\underline{\mathbf{F}}$, to represent the solution, and for $n^{arr} > 1$ it is therefore not able to accurately solve the problem.

Finite Queues

When queues have a finite buffer, contributions from eigenvalues outside the unit disc can also be normalised and need to be included in the solution process. As in the infinite case, the number of these depends on n^{arr}, the number of distinct arrival processes. In addition to these, n^{serv} and n^{kill}, the number of distinct service and killing processes need to be considered and we have $n^* = (n^{arr} + n^{serv} + n^{kill})N$ for the finite case. By way of forward and backward series, explained in section 3.6.2, the MG method can represent a total of $2N$ eigenvalues, so that a solution for a queue where $n^{arr} + n^{serv} + n^{kill} > 2$ is not possible in general.

In addition to this limitation, there is the practical obstacle that current matrix geometric methods are generally applied to third order matrix recurrence relations, leading to the solution of a quadratic matrix polynomial. Matrix analytic techniques seek the solution to a higher order problem, but still represent the solution using a single matrix, or pair of matrices (one forward and backward). It should be noted that the folding methods we describe in section 3.7 transform a system of higher order to a quadratic function, soluble using, for example, the Latouche-Ramaswami method [LatRam93]. The folding method is used to convert the problem into an equivalent one that has a complete matrix-geometric solution, but with some numerical stability issues. These

[7] These are either unbatched arrival processes (*i.e.* batch parameter 0), or an arrival process whose batch parameter is equal to that of another arrival process, and hence can be combined into one.

numerical issues are not caused by the folding process: they are inherent in raising matrices to large powers.

3.7 Folded Matrix geometric method

We will now present a transformation process that facilitates the MG solution process of a particular Markovian geometrically batched queue originally investigated in [ChaHar01]. The queue features 3 distinct, geometrically batched streams for each of its modulation states; one for each of positive and negative customers and processing completions. In [ChaHar01] the localised balance equations had been derived by hand. Using our automated approach [ThoZat+03], we have derived the following localised matrix recurrence of order 4 that holds within the repeating region

$$(\overrightarrow{\mathbf{v}}_j\underline{\mathbf{Q}}_0 + \overrightarrow{\mathbf{v}}_{j+1}\underline{\mathbf{Q}}_1 + \overrightarrow{\mathbf{v}}_{j+2}\underline{\mathbf{Q}}_2 + \overrightarrow{\mathbf{v}}_{j+3}\underline{\mathbf{Q}}_3) = \overrightarrow{\mathbf{0}} \quad \epsilon^b \leq j \leq \epsilon^t \tag{3.28}$$

Where, using the notation of section 2,

$$\begin{aligned}
\underline{\mathbf{Q}}_0 &= \underline{\mathbf{\Lambda}} + (\underline{\mathbf{K}} + \underline{\mathbf{M}} - \underline{\mathbf{Q}})\underline{\mathbf{\Theta}} \\
\underline{\mathbf{Q}}_1 &= \underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{\Delta}} + \underline{\mathbf{\Phi}})) - \underline{\mathbf{K}}(\underline{\mathbf{\Theta}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}) + \underline{\mathbf{I}}) \\
&\quad -\underline{\mathbf{M}}(\underline{\mathbf{\Theta}}(\underline{\mathbf{I}} + \underline{\mathbf{\Delta}}) + \underline{\mathbf{I}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{\Delta}} + \underline{\mathbf{I}}) \\
\underline{\mathbf{Q}}_2 &= \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{\Phi\Theta}}) + \underline{\mathbf{M}}(\underline{\mathbf{I}} + \underline{\mathbf{\Delta}} + \underline{\mathbf{\Delta\Theta}}) \\
&\quad +\underline{\mathbf{\Lambda}}(\underline{\mathbf{\Delta}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{\Delta\Phi}}) - \underline{\mathbf{Q}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{\Delta}} + \underline{\mathbf{\Delta\Theta\Phi}}) \\
\underline{\mathbf{Q}}_3 &= (\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}})\underline{\mathbf{\Delta\Phi}} - \underline{\mathbf{M\Delta}} - \underline{\mathbf{K\Phi}}
\end{aligned} \tag{3.29}$$

In the absence of degenerate streams, the number of eigenvalues in the repeating region is given by $n^* = (n^{arr} + n^{serv} + n^{kill})N$. We have $n^{arr} = n^{serv} = n^{kill} = 1$, so that in this example $n^* = 3N$. As has been seen before, the classic matrix geometric method is insufficient to represent all these.

3.7.1 Folding of a modulated queue

A queue with batched arrival or processing completion processes creates a transition structure which leads to Kolmogorov balance equations constrained over a range of

more than three queue lengths. Haverkort and Ost [HavOst97] make the observation that an M/M/1 queue with *fixed* batch sizes cannot be solved by matrix geometric methods, and suggest a means for re-formulating the solution. This involves "folding" the queue down on itself: the states for neighbouring queue lengths are re-cast to occupy the same level in the lattice of states, thereby increasing the number of modulation states and shortening the queue. We propose to apply the same methodology to our localised, geometric batch sizes.

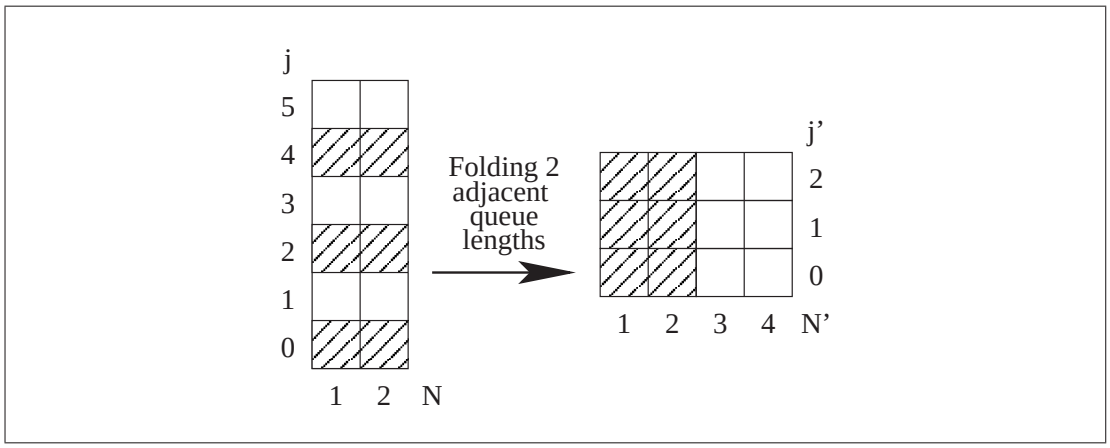


Figure 3.7: State correspondence for an example queue during the folding process.

Figure 3.7 shows how the states correspond between the original and the folded queue, for a queue with maximal queue length $L = 5$ and $N = 2$ modulation states. States at even queue lengths have been shaded to illustrate the folding effect: The folded level j is composed of the original levels $2j$ and $2j+1$. As a result the queue length is halved, but the number of modulation states is doubled, with the folded modulation states 1 and 2 corresponding to even queue lengths and states 3 and 4 to odd queue lengths in the original queue.

The forward map, going from the original to the folded queue is

$$(j,n) \mapsto \begin{cases} (\frac{j}{2}, n) & \text{if } j \text{ is even} \\ (\frac{j-1}{2}, n+2) & \text{if } j \text{ is odd} \end{cases} \tag{3.30}$$

To map a folded state, (j', n') back onto the original queue, we use

$$(j',n') \mapsto \begin{cases} (2j', n') & \text{if } n < 2 \\ (2j'+1, 1+n' \mod 2) & \text{if } n \geq 2 \end{cases} \tag{3.31}$$

Both of the mappings are well defined, because the number of queue lengths is divisi-

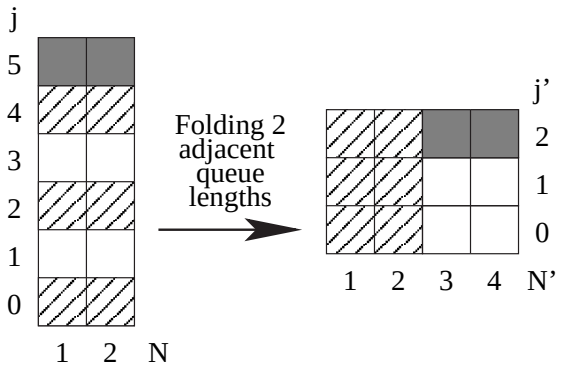


Figure 3.8: Folding using dummy states.

ble by our folding factor, 2. When this is not so, we need to first pad the original queue with a dummy state, as in figure 3.8, and then proceed with the folding process as before. Subsequent solution processes will then assign a zero-probability of residing within the dummy state, as there are no transitions entering it.

We do not have to fold a queue by a factor of 2, we can use any positive integer for this purpose. For this general case, when the folding factor is f, the forward and reverse maps are given as follows

$$(j,n) \mapsto \left(\frac{j-(j \mod f)}{f}, n+(j \mod f)f \right)$$

$$(j',n') \mapsto \left(fj' + \left\lfloor \frac{n'}{N} \right\rfloor, 1+n' \mod N \right)$$

The construction of the Kolmogorov balance equations for such a system is covered in our main example in section 3.8. The efficiency implications for both spectral expansion and matrix geometric methods in application to queues folded in this manner is also discussed in [HavOst97]. At this point we do not concern ourselves with efficiency: instead, we concentrate on demonstrating the procedure, originally suggested for fixed batch sizes, on our queue type with unbounded geometric batches.

3.7.2 Folding balance equations

Just as in the preceding simple example we will now apply the folding to our geometrically batched queue represented by the recurrence relation (3.28). For this class of queue, the localised balance equations involve more than three queue levels, but can also be folded. The number of adjacent levels that need to be grouped to obtain a recurrence relation of order 2 is $v = 2$. This gives $\overrightarrow{\mathbf{u}}_j = \left(\begin{array}{cc} \overrightarrow{\mathbf{v}}_{2j} & \overrightarrow{\mathbf{v}}_{2j+1} \end{array} \right)$. The localised balance equations address the balance of flux in and out of states at a single queue length. The composite states of the folded queue comprise a number of queue lengths, and the original balance equations are included once for each original level within a single compound, folded level. The folded recurrence relation thus becomes:

$$\overrightarrow{\mathbf{u}}_j \underline{\mathbf{A}}_0 + \overrightarrow{\mathbf{u}}_{j+1} \underline{\mathbf{A}}_1 + \overrightarrow{\mathbf{u}}_{j+2} \underline{\mathbf{A}}_2 = \overrightarrow{\mathbf{0}} \quad \lceil \tfrac{\epsilon^b}{2} \rceil \leq j \leq \lfloor \tfrac{\epsilon^t}{2} \rfloor \tag{3.32}$$

where

$$\underline{\mathbf{A}}_0 = \begin{pmatrix} \underline{\mathbf{Q}}_0 & \underline{\mathbf{0}} \\ \underline{\mathbf{Q}}_1 & \underline{\mathbf{Q}}_0 \end{pmatrix} \quad \underline{\mathbf{A}}_1 = \begin{pmatrix} \underline{\mathbf{Q}}_2 & \underline{\mathbf{Q}}_1 \\ \underline{\mathbf{Q}}_3 & \underline{\mathbf{Q}}_2 \end{pmatrix} \quad \underline{\mathbf{A}}_2 = \begin{pmatrix} \underline{\mathbf{0}} & \underline{\mathbf{Q}}_3 \\ \underline{\mathbf{0}} & \underline{\mathbf{0}} \end{pmatrix} \tag{3.33}$$

or, alternatively,

$$\underline{\mathbf{A}}_0 = \begin{pmatrix} \underline{\mathbf{0}} & \underline{\mathbf{0}} \\ \underline{\mathbf{Q}}_0 & \underline{\mathbf{0}} \end{pmatrix} \quad \underline{\mathbf{A}}_1 = \begin{pmatrix} \underline{\mathbf{Q}}_1 & \underline{\mathbf{Q}}_0 \\ \underline{\mathbf{Q}}_2 & \underline{\mathbf{Q}}_1 \end{pmatrix} \quad \underline{\mathbf{A}}_2 = \begin{pmatrix} \underline{\mathbf{Q}}_3 & \underline{\mathbf{Q}}_2 \\ \underline{\mathbf{0}} & \underline{\mathbf{Q}}_3 \end{pmatrix} \tag{3.34}$$

Note that for the first choice, the last matrix, $\underline{\mathbf{A}}_2$ is singular[8] whereas for the second it is the first matrix $\underline{\mathbf{A}}_0$ that is singular. Both choices are satisfactory and will cause N zero-eigenvalues to be introduced into either the solution $\hat{\underline{\mathbf{B}}}$ to the backward series or into that of the forward series, $\hat{\underline{\mathbf{F}}}$. We will be using the values for $\underline{\mathbf{A}}_i$ given in (3.33).

$$(\underline{\mathbf{A}}_0 + \underline{\mathbf{F}}\,\underline{\mathbf{A}}_1 + \underline{\mathbf{F}}^2 \underline{\mathbf{A}}_2) = \underline{\mathbf{0}}$$

and

$$(\underline{\mathbf{A}}_2 + \underline{\mathbf{B}}\,\underline{\mathbf{A}}_1 + \underline{\mathbf{B}}^2 \underline{\mathbf{A}}_0) = \underline{\mathbf{0}}$$

Solving the matrix recurrences using the Logarithmic Reduction (LR) method yields the required solution matrices $\hat{\underline{\mathbf{F}}}$ and $\hat{\underline{\mathbf{B}}}$, both of which are of sufficient size to allow

[8] Later in section 3.8.7 we use a system in which $\underline{\mathbf{A}}_2$ is non-singular.

for $4N$ eigenvalues and eigenvectors. N of the eigenvalues must be zero because the zero-subspaces of $\underline{\mathbf{A}}_0$ (or $\underline{\mathbf{A}}_2$) is of dimension N. Other zero-eigenvalues can certainly also appear depending on the parameters used for the various rate and batch matrices.

3.8 Numerical Examples

We will first look at the solution of a finite queue of the basic form given in the previous section, and an infinite queue with an additional independent arrival process (corresponding to a queue in a network with arrivals from two other queues, for example). These particular examples are chosen to allow us to show concrete examples of circumstances in which it is not possible to solve using the original MG method. We then use the folded queue expression derived in the previous section to provide MG with a spectrum of sufficient size to accommodate the solution.

For simplicity, we use queues with two modulation states. The total number of eigenvalues in the solution to such queues is found, in practice and easily verified by examination of the determinant as in section 3.4.1, to be equal to the number of modulation states, N, multiplied by the number of independent geometrically batched arrival and departure processes. The finite system we examine therefore has 6 eigenvalues, all of which are necessary to the solution. The infinite system has 8 eigenvalues, 4 of which lie within the unit disc.

3.8.1 Finite Queue Example

We consider a particular instance of this queue, characterised by the following stream parameters.

$$\underline{\mathbf{\Lambda}} = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \quad \underline{\mathbf{\Theta}} = \begin{pmatrix} \frac{3}{10} & 0 \\ 0 & \frac{1}{5} \end{pmatrix} \quad \underline{\mathbf{M}} = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

$$\underline{\mathbf{\Phi}} = \begin{pmatrix} \frac{3}{10} & 0 \\ 0 & \frac{1}{10} \end{pmatrix} \quad \underline{\mathbf{K}} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \quad \underline{\mathbf{R}} = \begin{pmatrix} \frac{2}{5} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

and modulation matrix

$$\underline{\mathbf{Q}} = \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix}$$

In addition, we use a single processor. The killing paradigm we employ in this example removes jobs from the tail of the queue, including those in service[9], and the waiting room of the queue is finite and of even size, *i.e.* queue length $0 \ldots L$, where L is odd. Using the expressions given in (3.29) these values provide unique values for the $\underline{\mathbf{Q}}_i$ matrices, which we will now use in solving for the steady-state solution of the repeating region — first, using spectral expansion and then using matrix geometric methods with folding.

3.8.2 Spectral Expansion approach

As $\underline{\mathbf{Q}}_3$ is non-singular, we are able to form the quantities $-\underline{\mathbf{Q}}_i\underline{\mathbf{Q}}_3^{-1}$ and use these to populate the matrix $\underline{\mathbf{A}}$ as given in (3.7).

$$\underline{\mathbf{A}} = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{6967}{3001} & -\frac{219}{6002} \\ 0 & 0 & 0 & 0 & -\frac{348}{3001} & \frac{5603}{3001} \\ 1 & 0 & 0 & 0 & -\frac{184647}{30010} & \frac{21199}{60020} \\ 0 & 1 & 0 & 0 & \frac{2482}{3001} & -\frac{33425}{6002} \\ 0 & 0 & 1 & 0 & \frac{139089}{30010} & -\frac{9871}{60020} \\ 0 & 0 & 0 & 1 & \frac{4772}{15005} & \frac{131967}{30010} \end{pmatrix}$$

The 6 eigenvalues of this matrix are evaluated numerically and are given below to 8 digits precision.

$$\lambda^{SE} = \{0.49281008, 0.84141200, 1, 1.63072196, 2.40960738, 2.65763786\}$$

Note in particular the eigenvalues which are equal to, or greater than 1. Theoretical analysis of the Latouche-Ramaswami and other related methods suggests that the matrix should contain only those eigenvalues less than 1 [BinLat+02]. Therefore, our

[9]The choice of killing paradigm affects the boundary conditions at the empty queue, and we use the simplest here

results contradict a common assumption of the matrix geometric method. There are no repeated eigenvalues so that each eigenvalue corresponds to a distinct eigenvector, the set of which is linearly independent. Using the projection matrix given in (3.14), the eigenvectors are given as

$$\overrightarrow{\boldsymbol{\zeta}}^{SE} = \begin{pmatrix} -0.58750 & -0.90188 & -0.89442 & -0.81371 & -0.07298 & -0.92778 \\ 0.809222 & -0.43198 & -0.44721 & 0.58125 & 0.99733 & 0.37312 \end{pmatrix}$$

3.8.3 Matrix Geometric approach

Using the expressions gained for $\underline{\mathbf{Q}}_i$, we apply the folding methodology as introduced in section 3.7 and evaluate the numerical solutions $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}$ to the forward and backward series formed by the matrices $\underline{\mathbf{A}}_0$, $\underline{\mathbf{A}}_1$ and $\underline{\mathbf{A}}_2$. We find

$$\underline{\hat{\mathbf{F}}} = \begin{pmatrix} -0.88033194 & -0.01294500 & -1.75444663 & 0.05136837 \\ 0.02704049 & -0.76770941 & 0.25351728 & -1.51335404 \\ 1.99675926 & -0.08729268 & 3.12940253 & -0.35905559 \\ -0.25989511 & 1.96818477 & -1.00342909 & 3.12872885 \end{pmatrix}$$

$$\underline{\hat{\mathbf{B}}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.37509951 & 0.16049028 & 0.14065278 & 0.12699264 \\ -0.002920315 & 0.41617984 & -0.00231079 & 0.17315788 \end{pmatrix}$$

The eigenvalues of $\underline{\hat{\mathbf{F}}}$ are $\lambda^{\hat{F}} = \{0.24286178, 0.70797415, 1, 2.65925410\}$ and those of $\underline{\hat{\mathbf{B}}}$ are $\lambda^{\hat{B}} = \{0.17222946, 0.14158212, 0, 0\}$. To relate these to those found by the spectral expansion method, we note that our folding methodology grouped pairs of levels together. A multiplication by $\underline{\hat{\mathbf{F}}}$ now relates the probability vector for level j, $\overrightarrow{\mathbf{u}}_j = \left(\overrightarrow{\mathbf{v}}_{2j}, \overrightarrow{\mathbf{v}}_{2j+1}\right)$, to the one at level $j+1$, $\overrightarrow{\mathbf{u}}_{j+1} = \left(\overrightarrow{\mathbf{v}}_{2j+2}, \overrightarrow{\mathbf{v}}_{2j+3}\right)$, so that one multiplication within the folded representation gives a jump of 2 levels instead of just one. We therefore need to raise each eigenvalue to the power[10] $1/2$ to compare them with those found when using the system without folding. In addition, the matrix $\underline{\hat{\mathbf{B}}}$ relates the vector $\overrightarrow{\mathbf{u}}_j$ to $\overrightarrow{\mathbf{u}}_{j-1}$, a relation that runs in the opposite direction to that for

[10] For k folded levels, use power $1/k$

$\underline{\hat{\mathbf{F}}}$ and the eigenmodes used in spectral expansion. If $\overrightarrow{\mathbf{u}}_j\underline{\hat{\mathbf{B}}} = \overrightarrow{\mathbf{u}}_{j-1}$, then (if $\underline{\hat{\mathbf{B}}}$ is non-singular) $\overrightarrow{\mathbf{u}}_{j-1}\underline{\hat{\mathbf{B}}}^{-1} = \overrightarrow{\mathbf{u}}_j$ is an equivalent relation that produces probabilities at *increasing* queue lengths. Thus, we need to calculate the eigenvalues and eigenvectors of $\underline{\hat{\mathbf{B}}}^{-1}$. We cannot form the inverse as two of $\underline{\hat{\mathbf{B}}}$'s eigenvalues are zero and it is therefore singular. We do, however know that the inverse of the linear transformation that the matrix $\underline{\hat{\mathbf{B}}}$ performs on the subspace that is spanned by its non-zero eigenmodes, is given by the reciprocal of the (non-zero) eigenvalues of $\underline{\hat{\mathbf{B}}}$ as well as the eigenvectors of $\underline{\hat{\mathbf{B}}}$:

$$\begin{aligned}\sqrt{\lambda^{\hat{F}}} &= \{0.49281008, 0.84141200, 1, 1.63072196\}\\ \frac{1}{\sqrt{\lambda^{\hat{B}}}} &= \{2.40960738, 2.65763786\}\end{aligned}$$

exactly as in spectral expansion.

Again, note that the set $\lambda^{\hat{F}}$ contains eigenvalues of magnitude larger than one, so that the assertion in [BinLat+02] that eigenvalues are within the unit disc does not hold in this case.

The eigenvectors for the non-zero eigenvalues of both matrices are of the form $\overrightarrow{\boldsymbol{\psi}}_i^{\hat{F}} = (\overrightarrow{\boldsymbol{\zeta}}_i^{\hat{F}}, \sqrt{\lambda_i^{\hat{F}}}\overrightarrow{\boldsymbol{\zeta}}_i^{\hat{F}})$ and $\overrightarrow{\boldsymbol{\psi}}_i^{\hat{B}} = (\overrightarrow{\boldsymbol{\zeta}}_i^{\hat{B}}, \sqrt{\lambda_i^{\hat{B}}}\overrightarrow{\boldsymbol{\zeta}}_i^{\hat{B}})$. In addition, the matrix $\underline{\hat{\mathbf{B}}}$ has the two eigenvectors $(1,0,0,0)$ and $(0,1,0,0)$ associated with its two zero eigenvalues. Using a projection matrix (3.14) on the $\overrightarrow{\boldsymbol{\psi}}$'s, we find that

$$\overrightarrow{\boldsymbol{\zeta}}^{\hat{F}} = \begin{pmatrix} 0.52698282 & -0.69009480 & -0.63245553 & 0.42538088 \\ -0.72586762 & -0.33054215 & -0.31622777 & -0.30385991 \end{pmatrix}$$

and

$$\overrightarrow{\boldsymbol{\zeta}}^{\hat{B}} = \begin{pmatrix} 0.06741043 & -0.86834332 \\ -0.92115765 & 0.34922364 \end{pmatrix}$$

all of which are parallel to the eigenvectors corresponding to the same eigenvalue in $\overrightarrow{\boldsymbol{\zeta}}^{SE}$.

Comparison with the spectral expansion approach shows that $\underline{\hat{\mathbf{F}}}$ contains the spectral information for the smallest 4 eigenvalues, whereas $\underline{\hat{\mathbf{B}}}$ holds the information for the remaining 2 eigenmodes, as well as the two additional zero eigenvalues due to the kernel of $\underline{\mathbf{A}}_2$ in (3.33).

3.8.4 Boundary conditions for a finite queue

So far we have derived expressions for the relative probability distributions within the repeating region. For spectral expansion, we have

$$\overrightarrow{\mathbf{v}}_j = \sum_{k=1}^{6} \alpha_k \left(\lambda_k^{SE}\right)^j \overrightarrow{\boldsymbol{\zeta}}_k^{SE} \quad 1 \leq j \leq L-1$$

for the folded matrix geometric representation, we write

$$\overrightarrow{\mathbf{w}}_j = \overrightarrow{\mathbf{f}} . \underline{\hat{\mathbf{F}}}^j + \overrightarrow{\mathbf{b}} . \underline{\hat{\mathbf{B}}}^{\frac{L-1}{2}-j} \quad 1 \leq j \leq \tfrac{L-1}{2} - 1$$

The probabilities outside the repeating region are represented by explicit unknowns $\overrightarrow{\mathbf{v}}_0$ and $\overrightarrow{\mathbf{v}}_L$ ($\overrightarrow{\mathbf{w}}_0$ and $\overrightarrow{\mathbf{w}}_{\frac{L-1}{2}}$). To constrain these as well as the free parameters given by $\{\alpha_1, \ldots, \alpha_6\}$ or $\{\overrightarrow{\mathbf{f}}, \overrightarrow{\mathbf{b}}\}$, we include those balance equations in the solution process which differ from the one (3.29) that holds within the repeating region. For our queue, these levels are $j \in \{0, 1, L-2, L-1, L\}$. When folding is employed, the corresponding levels are $j \in \{0, \frac{L-1}{2} - 1, \frac{L-1}{2}\}$.

3.8.5 Not folded

The symbolic expressions for the boundary equations when using spectral expansion are given below.

Level 0

$$\begin{aligned}
&\overrightarrow{\mathbf{v}}_0.(\underline{\mathbf{Q}}(\underline{\mathbf{R}} - \underline{\mathbf{I}})(\underline{\boldsymbol{\Theta}} - \underline{\mathbf{I}}) - \underline{\boldsymbol{\Lambda}}(\underline{\mathbf{R}\boldsymbol{\Theta}} - \underline{\mathbf{I}})(\underline{\boldsymbol{\Phi}\boldsymbol{\Theta}} - \underline{\mathbf{I}}) + \\
&\overrightarrow{\mathbf{v}}_1.(\underline{\mathbf{Q}}(\underline{\mathbf{R}\boldsymbol{\Phi}} - \underline{\mathbf{R}} - \underline{\boldsymbol{\Phi}}) + \underline{\mathbf{K}} + \underline{\mathbf{M}} - \underline{\boldsymbol{\Lambda}}(\underline{\mathbf{R}\boldsymbol{\Theta}\boldsymbol{\Phi}} - \underline{\mathbf{R}} - \underline{\boldsymbol{\Phi}})) + \\
&\overrightarrow{\mathbf{v}}_2.((\underline{\mathbf{Q}} + \underline{\boldsymbol{\Lambda}})\underline{\mathbf{R}\boldsymbol{\Phi}} - \underline{\mathbf{K}\boldsymbol{\Phi}} - \underline{\mathbf{M}\mathbf{R}}) = \overrightarrow{\mathbf{0}}
\end{aligned}$$

Level 1

$$\begin{aligned}
&\overrightarrow{\mathbf{v}}_0.(\underline{\boldsymbol{\Lambda}}(\underline{\mathbf{I}} - \underline{\boldsymbol{\Theta}})(\underline{\mathbf{R}\boldsymbol{\Theta}} - \underline{\mathbf{I}})(\underline{\boldsymbol{\Theta}\boldsymbol{\Phi}} - \underline{\mathbf{I}})) + \\
&\overrightarrow{\mathbf{v}}_1.(\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{M}} - \underline{\boldsymbol{\Lambda}}((\underline{\boldsymbol{\Theta}} - \underline{\mathbf{I}})(\underline{\boldsymbol{\Phi}} - \underline{\mathbf{R}\boldsymbol{\Theta}\boldsymbol{\Phi}} + \underline{\mathbf{R}}) - \underline{\mathbf{I}})) + \\
&\overrightarrow{\mathbf{v}}_2.(\underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\boldsymbol{\Phi}}) + \underline{\mathbf{M}}(\underline{\mathbf{I}} + \underline{\mathbf{R}}) - \underline{\mathbf{Q}}(\underline{\mathbf{R}} + \underline{\boldsymbol{\Phi}}) + \underline{\boldsymbol{\Lambda}}(\underline{\mathbf{R}} + \underline{\boldsymbol{\Phi}} + \underline{\mathbf{R}\boldsymbol{\Phi}} - \underline{\mathbf{R}\boldsymbol{\Theta}\boldsymbol{\Phi}})) + \\
&\overrightarrow{\mathbf{v}}_3.(\underline{\mathbf{Q}\mathbf{R}\boldsymbol{\Phi}} - \underline{\mathbf{K}\boldsymbol{\Phi}} - \underline{\mathbf{M}\mathbf{R}} - \underline{\boldsymbol{\Lambda}\mathbf{R}\boldsymbol{\Phi}}) = \overrightarrow{\mathbf{0}}
\end{aligned}$$

Level $L-2$

$$\begin{aligned}
&\overrightarrow{\mathbf{v}}_{L-3}.(\underline{\mathbf{\Theta}}(\underline{\mathbf{K}}+\underline{\mathbf{M}})-\underline{\mathbf{Q}}\underline{\mathbf{\Theta}}+\underline{\mathbf{\Lambda}})+\\
&\overrightarrow{\mathbf{v}}_{L-2}.(\underline{\mathbf{Q}}(\underline{\mathbf{I}}+\underline{\mathbf{\Theta}}(\underline{\mathbf{\Phi}}+\underline{\mathbf{R}}-\underline{\mathbf{I}}))-\underline{\mathbf{K}}(\underline{\mathbf{I}}+\underline{\mathbf{\Theta}\mathbf{\Phi}})-\underline{\mathbf{M}}(\underline{\mathbf{I}}+\underline{\mathbf{R}\mathbf{\Theta}})-\underline{\mathbf{\Lambda}}(\underline{\mathbf{R}}+\underline{\mathbf{\Phi}}))+\\
&\overrightarrow{\mathbf{v}}_{L-1}.(\underline{\mathbf{Q}}((\underline{\mathbf{\Theta}}-\underline{\mathbf{I}})(\underline{\mathbf{\Phi}}+\underline{\mathbf{R}}-\underline{\mathbf{I}})-\underline{\mathbf{R}\mathbf{\Theta}\mathbf{\Phi}})+\underline{\mathbf{K}\mathbf{\Phi}}+\underline{\mathbf{M}\mathbf{R}}+\underline{\mathbf{\Lambda}\mathbf{R}\mathbf{\Phi}})+\\
&\overrightarrow{\mathbf{v}}_{L}.(\underline{\mathbf{Q}}(\underline{\mathbf{I}}-\underline{\mathbf{R}})(\underline{\mathbf{\Theta}}-\underline{\mathbf{I}})(\underline{\mathbf{\Phi}}-\underline{\mathbf{I}}))=\overrightarrow{\mathbf{0}}
\end{aligned}$$

Level $L-1$

$$\begin{aligned}
&\overrightarrow{\mathbf{v}}_{L-2}.(\underline{\mathbf{\Theta}}(\underline{\mathbf{K}}+\underline{\mathbf{M}})-\underline{\mathbf{Q}}\underline{\mathbf{\Theta}}+\underline{\mathbf{\Lambda}})+\\
&\overrightarrow{\mathbf{v}}_{L-1}.(\underline{\mathbf{Q}}(\underline{\mathbf{I}}+(\underline{\mathbf{R}}-\underline{\mathbf{I}})\underline{\mathbf{\Theta}})-\underline{\mathbf{K}}-\underline{\mathbf{M}}(\underline{\mathbf{I}}+\underline{\mathbf{R}\mathbf{\Theta}}-\underline{\mathbf{\Theta}\mathbf{\Phi}})-\underline{\mathbf{\Lambda}\mathbf{R}})+\\
&\overrightarrow{\mathbf{v}}_{L}.(\underline{\mathbf{Q}}(\underline{\mathbf{R}}-\underline{\mathbf{I}})(\underline{\mathbf{\Theta}}-\underline{\mathbf{I}})+\underline{\mathbf{M}}(\underline{\mathbf{R}}-\underline{\mathbf{\Phi}})(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}\mathbf{\Phi}}))=\overrightarrow{\mathbf{0}}
\end{aligned}$$

Level L

$$\begin{aligned}
&\overrightarrow{\mathbf{v}}_{L-1}.(\underline{\mathbf{\Theta}}(\underline{\mathbf{K}}+\underline{\mathbf{M}})-\underline{\mathbf{Q}}\underline{\mathbf{\Theta}}+\underline{\mathbf{\Lambda}})+\\
&\overrightarrow{\mathbf{v}}_{L}.(\underline{\mathbf{Q}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})+\underline{\mathbf{K}}(\underline{\mathbf{R}\mathbf{\Theta}}-\underline{\mathbf{I}})+\underline{\mathbf{M}}(\underline{\mathbf{\Theta}\mathbf{\Phi}}-\underline{\mathbf{I}}))=\overrightarrow{\mathbf{0}}
\end{aligned}$$

3.8.6 Folded for matrix geometric solution

For the matrix geometric case which necessitates folding, we write $\underline{\mathbf{Q}}_k^j$ as the coefficient of $\overrightarrow{\mathbf{v}}_k$ in the balance equation for level j, so that *e.g.* $\underline{\mathbf{Q}}_{L-1}^L=\underline{\mathbf{\Theta}}(\underline{\mathbf{K}}+\underline{\mathbf{M}}-\underline{\mathbf{Q}})+\underline{\mathbf{\Lambda}}$ as given in the previous section. The folded boundary equations for the various levels are now for

Level 0 & 1:

$$\overrightarrow{\mathbf{w}}_0.\begin{pmatrix}\underline{\mathbf{Q}}_0^0 & \underline{\mathbf{Q}}_0^1\\ \underline{\mathbf{Q}}_1^0 & \underline{\mathbf{Q}}_1^1\end{pmatrix}+\overrightarrow{\mathbf{w}}_1.\begin{pmatrix}\underline{\mathbf{Q}}_2^0 & \underline{\mathbf{Q}}_2^1\\ \underline{\mathbf{0}} & \underline{\mathbf{Q}}_3^1\end{pmatrix}=\overrightarrow{\mathbf{0}}$$

Level $L-2$:

$$\overrightarrow{\mathbf{w}}_{\frac{L-1}{2}-1}.\begin{pmatrix}\underline{\mathbf{Q}}_{L-3}^{L-2}\\ \underline{\mathbf{Q}}_{L-2}^{L-2}\end{pmatrix}+\overrightarrow{\mathbf{w}}_{\frac{L-1}{2}}.\begin{pmatrix}\underline{\mathbf{Q}}_{L-1}^{L-2}\\ \underline{\mathbf{Q}}_{L}^{L-2}\end{pmatrix}=\overrightarrow{\mathbf{0}}$$

Level $L-1$ & L:

$$\overrightarrow{\mathbf{w}}_{\frac{L-1}{2}-1}.\begin{pmatrix}\underline{\mathbf{0}} & \underline{\mathbf{0}}\\ \underline{\mathbf{Q}}_{L-2}^{L-1} & \underline{\mathbf{0}}\end{pmatrix}+\overrightarrow{\mathbf{w}}_{\frac{L-1}{2}}.\begin{pmatrix}\underline{\mathbf{Q}}_{L-1}^{L-1} & \underline{\mathbf{Q}}_{L-1}^{L}\\ \underline{\mathbf{Q}}_{L}^{L-1} & \underline{\mathbf{Q}}_{L}^{L}\end{pmatrix}=\overrightarrow{\mathbf{0}}$$

Figure 3.10 shows the probability distribution that the queue takes when $L=15$. Section 3.9 discusses the numerical accuracy in solving this system for a range of other (larger) L.

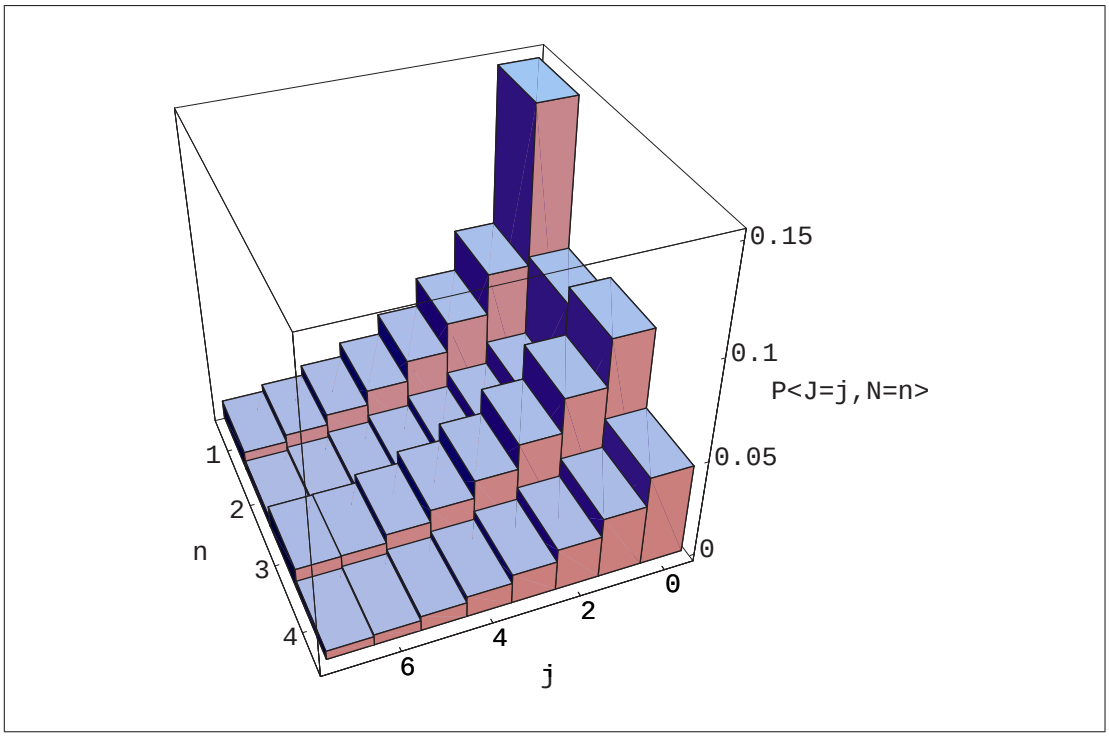


Figure 3.9: Finite queue example: steady-state probability distribution of the *folded* queue. Modulation states 1 and 2 correspond to even levels in the original queue, states 3 and 4 to the odd levels.

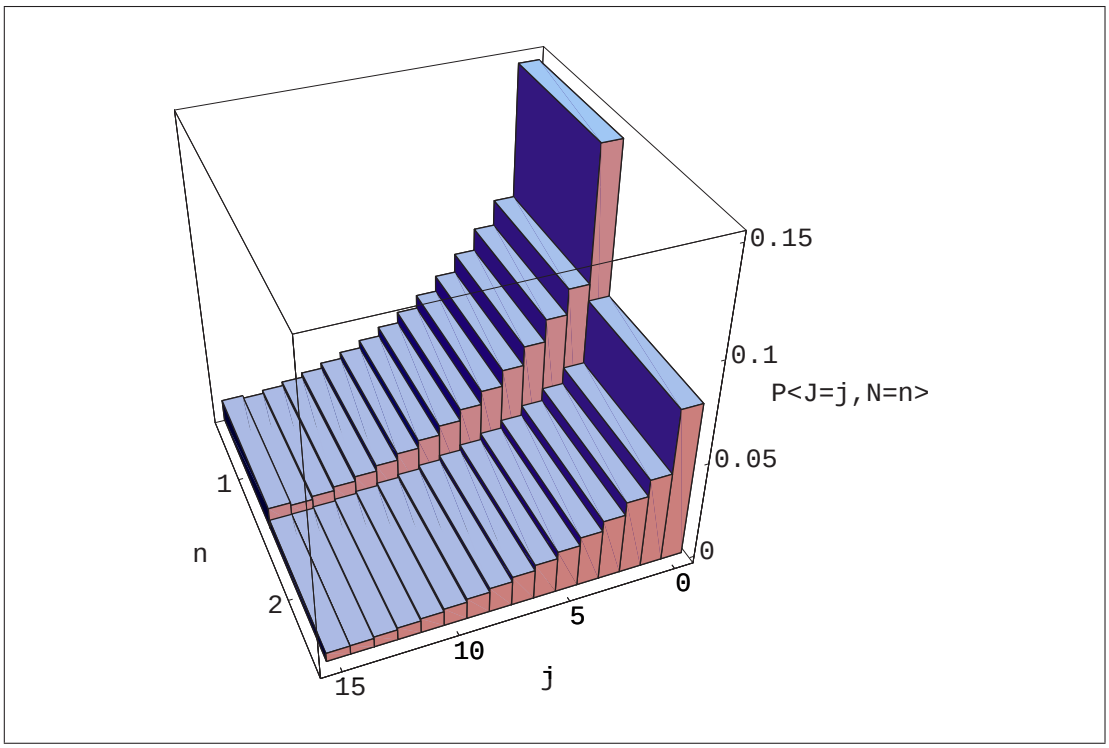


Figure 3.10: Finite queue example: steady-state probability distribution found using spectral expansion or the matrix geometric solution with unfolding.

3.8.7 Infinite Queue Example

When solving for the steady-state of infinite queues, the coefficients of eigenvalues on or outside the unit disc must be zero, as they cannot be normalised. The repeating-region recurrence relations — and with it all its eigenvalues and eigenvectors — are identical to those found in the finite case. There are two eigenvalues within the unit disc, so that folding is not necessary, because the single solution matrix $\underline{\hat{\mathbf{F}}}$ of the unfolded case is of sufficient size and contains both.

In practice, we require the ability to superpose arrival processes for queues in networks. To demonstrate the utility of folding for infinite queues, we allocate an infinite waiting room and add an arrival stream to augment the previous model which is given by rate matrix $\underline{\mathbf{\Lambda}}_2$ and batch matrix Θ_2 as follows:

$$\underline{\mathbf{\Lambda}}_2 = \begin{pmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{9} \end{pmatrix} \quad \underline{\mathbf{\Theta}}_2 = \begin{pmatrix} \frac{1}{10} & 0 \\ 0 & \frac{1}{20} \end{pmatrix}$$

We also re-label $\underline{\mathbf{\Lambda}} \mapsto \underline{\mathbf{\Lambda}}_1$ and $\underline{\mathbf{\Theta}} \mapsto \underline{\mathbf{\Theta}}_1$. The resulting symbolic form of the recurrence relation for the repeating region derived using methods in Appendix A is given below

$$\begin{aligned}
\underline{\mathbf{Q}}_0 = \;& \underline{\mathbf{Q}}\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Theta}}_2 - ((\underline{\mathbf{K}} + \underline{\mathbf{M}})\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Lambda}}_1) - \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Lambda}}_2 \\
\underline{\mathbf{Q}}_1 = \;& \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_2 + \underline{\mathbf{R}}\underline{\mathbf{\Theta}}_2 + \underline{\mathbf{\Phi}}\underline{\mathbf{\Theta}}_2) + \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{R}}\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Phi}}\underline{\mathbf{\Theta}}_1 \\
& + \underline{\mathbf{M}}(\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_2 + (\underline{\mathbf{I}} + \underline{\mathbf{R}})\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Theta}}_2) + \underline{\mathbf{K}}(\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_2 + (\underline{\mathbf{I}} + \underline{\mathbf{\Phi}})\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Theta}}_2) \\
& + \underline{\mathbf{Q}}(\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_2 + (\underline{\mathbf{\Phi}} + \underline{\mathbf{R}})\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Theta}}_2)) \\
\underline{\mathbf{Q}}_2 = \;& \underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}}) + (\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}}\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}})\underline{\mathbf{\Theta}}_2) \\
& - \underline{\mathbf{M}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{R}}\underline{\mathbf{\Theta}} + (\underline{\mathbf{I}} + \underline{\mathbf{R}} + \underline{\mathbf{R}}\underline{\mathbf{\Theta}}_1)\underline{\mathbf{\Theta}}_2) \\
& - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}} + (\underline{\mathbf{I}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}})\underline{\mathbf{\Theta}}_2) \\
& - \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{I}} + \underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + (\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}})\underline{\mathbf{\Theta}}_2) \\
& - \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}} + (\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}})\underline{\mathbf{R}}) \\
\underline{\mathbf{Q}}_3 = \;& \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_2)) + \underline{\mathbf{M}}(\underline{\mathbf{I}} + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_2)) \\
& - \underline{\mathbf{Q}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}(\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_2))) + \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_2) \\
& \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_1) \\
\underline{\mathbf{Q}}_4 = \;& \underline{\mathbf{Q}}\underline{\mathbf{R}\mathbf{\Phi}} - \underline{\mathbf{K}\mathbf{\Phi}} - \underline{\mathbf{M}\mathbf{R}} - \underline{\mathbf{\Lambda}}_1\underline{\mathbf{R}\mathbf{\Phi}} - \underline{\mathbf{\Lambda}}_2\underline{\mathbf{R}\mathbf{\Phi}}
\end{aligned}$$

Its order is one higher than before and it has a total of $4N = 8$ eigenvalues, 4 of which are within the unit disc. To ensure a matrix geometric solution with all 4 of the relevant

eigenmodes, we fold the queue as before, *i.e.* we use $\underline{\mathbf{A}}_0$, $\underline{\mathbf{A}}_1$ and $\underline{\mathbf{A}}_2$ as given below and solve the resulting second-order recurrence relation using the LR-algorithm.

$$\underline{\mathbf{A}}_0 = \begin{pmatrix} \underline{\mathbf{Q}}_0 & \underline{\mathbf{0}} \\ \underline{\mathbf{Q}}_1 & \underline{\mathbf{Q}}_0 \end{pmatrix} \quad , \underline{\mathbf{A}}_1 = \begin{pmatrix} \underline{\mathbf{Q}}_2 & \underline{\mathbf{Q}}_1 \\ \underline{\mathbf{Q}}_3 & \underline{\mathbf{Q}}_2 \end{pmatrix} \quad , \underline{\mathbf{A}}_2 = \begin{pmatrix} \underline{\mathbf{Q}}_4 & \underline{\mathbf{Q}}_3 \\ \underline{\mathbf{0}} & \underline{\mathbf{Q}}_4 \end{pmatrix}$$

We find

$$\underline{\hat{\mathbf{F}}} = \begin{pmatrix} -0.0894858330 & -0.0175654602 & -0.0829948779 & -0.0243799599 \\ -0.00294677564 & -0.0304039902 & -0.0075404771 & -0.0198735998 \\ 0.895824252 & 0.146933042 & 0.736698742 & 0.208011207 \\ 0.161185554 & 0.639410188 & 0.244510838 & 0.402124882 \end{pmatrix}$$

The sets of eigenvalues $\lambda^{\hat{F}}$ and eigenvectors $\overrightarrow{\psi}^{\hat{F}} = (\overrightarrow{\zeta}^{\hat{F}}, \sqrt{\lambda^{\hat{F}}}\overrightarrow{\zeta}^{\hat{F}})$ of $\underline{\hat{\mathbf{F}}}$ satisfy

$$\sqrt{\lambda^{\hat{F}}} = \begin{pmatrix} 0.054104017 & 0.11388376 & 0.49875131 & 0.86849535 \end{pmatrix}$$

$$\overrightarrow{\zeta}^{\hat{F}} = \begin{pmatrix} 0.68071170 & -0.51971510 & -0.99318108 & -0.1300806 \\ 0.32659553 & 0.72848827 & -0.028069268 & -0.9900304 \end{pmatrix}$$

These eigenvalue and eigenvector solutions are again entirely consistent with those found using spectral expansion.

3.8.8 Boundary conditions for an infinite queue

The queue being infinite, there are no boundary equations at the top of the queue. The integration of the level 0, $\overrightarrow{\mathbf{w}}_0 = (\overrightarrow{\mathbf{v}}_0, \overrightarrow{\mathbf{v}}_1)$, with the repeating region $\overrightarrow{\mathbf{w}}_j = \overrightarrow{\mathbf{f}}\,\underline{\hat{\mathbf{F}}}^j$ is brought about by the inclusion of balance equations for the folded levels 0 and 1 (unfolded levels 0 through 2).

level 0

$$\begin{aligned} \overrightarrow{\mathbf{v}}_0. \quad & (\underline{\mathbf{Q}}(\underline{\mathbf{R}} - \underline{\mathbf{I}})(\underline{\mathbf{\Theta}}_1 - \underline{\mathbf{I}}) + \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{I}} - \underline{\mathbf{R}\mathbf{\Theta}}_1)(\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}} - \underline{\mathbf{I}}) \\ & + \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{I}} - \underline{\mathbf{R}\mathbf{\Theta}}_2)(\underline{\mathbf{\Theta}}_2\underline{\mathbf{\Phi}} - \underline{\mathbf{I}})) \\ + \overrightarrow{\mathbf{v}}_1. \quad & (\underline{\mathbf{Q}}(\underline{\mathbf{R}}(\underline{\mathbf{\Phi}} - \underline{\mathbf{I}}) - \underline{\mathbf{\Phi}}) + \underline{\mathbf{K}} + \underline{\mathbf{M}} + \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} - \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_1) \\ & + \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} - \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_2)) \\ + \overrightarrow{\mathbf{v}}_2. \quad & (\underline{\mathbf{Q}\mathbf{\Phi}\mathbf{R}} - \underline{\mathbf{\Phi}}(\underline{\mathbf{K}} + \underline{\mathbf{\Lambda}}_1\underline{\mathbf{R}} + \underline{\mathbf{\Lambda}}_2\underline{\mathbf{R}}) - \underline{\mathbf{M}\mathbf{R}}) = \overrightarrow{\mathbf{0}} \end{aligned}$$

level 1

$$\begin{aligned}
\overrightarrow{\mathbf{v}}_0. \quad & (\underline{\mathbf{\Lambda}}_1(\underline{\mathbf{\Theta}}_1 - \underline{\mathbf{I}})(\underline{\mathbf{R}\mathbf{\Theta}}_1 - \underline{\mathbf{I}})(\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}} - \underline{\mathbf{I}}) \\
& +\underline{\mathbf{\Lambda}}_2(\underline{\mathbf{\Theta}}_2 - \underline{\mathbf{I}})(\underline{\mathbf{R}\mathbf{\Theta}}_2 - \underline{\mathbf{I}})(\underline{\mathbf{\Theta}}_2\underline{\mathbf{\Phi}} - \underline{\mathbf{I}})) \\
+\overrightarrow{\mathbf{v}}_1. \quad & (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{M}} - \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{I}} + (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}_1)\underline{\mathbf{\Phi}} + \underline{\mathbf{R}}(\underline{\mathbf{\Theta}}_1 - \underline{\mathbf{I}})(\underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}} - \underline{\mathbf{I}})) \\
& -\underline{\mathbf{\Lambda}}_2(\underline{\mathbf{I}} + (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}_2)\underline{\mathbf{\Phi}} + \underline{\mathbf{R}}(\underline{\mathbf{\Theta}}_2 - \underline{\mathbf{I}})(\underline{\mathbf{\Theta}}_2\underline{\mathbf{\Phi}} - \underline{\mathbf{I}}))) \\
+\overrightarrow{\mathbf{v}}_2. \quad & (\underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}) + \underline{\mathbf{M}}(1 + \underline{\mathbf{R}}) - \underline{\mathbf{Q}}(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}}) + \underline{\mathbf{\Lambda}}_1(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}} - \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_1 \\
& +\underline{\mathbf{\Lambda}}_2(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}} - \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_2))) \\
+\overrightarrow{\mathbf{v}}_3. \quad & (\underline{\mathbf{Q}\mathbf{\Phi}\mathbf{R}} - \underline{\mathbf{\Phi}}(\underline{\mathbf{K}} + \underline{\mathbf{\Lambda}}_1\underline{\mathbf{R}} + \underline{\mathbf{\Lambda}}_2\underline{\mathbf{R}}) - \underline{\mathbf{M}\mathbf{R}}) = \overrightarrow{\mathbf{0}}
\end{aligned}$$

level 2

$$\begin{aligned}
\overrightarrow{\mathbf{v}}_0. \quad & (\underline{\mathbf{\Lambda}}_2(\underline{\mathbf{\Theta}}_1 - \underline{\mathbf{\Theta}}_2)(\underline{\mathbf{\Theta}}_2 - \underline{\mathbf{I}})(\underline{\mathbf{R}\mathbf{\Theta}}_2 - \underline{\mathbf{I}})(\underline{\mathbf{\Phi}\mathbf{\Theta}}_2 - \underline{\mathbf{I}})) \\
+\overrightarrow{\mathbf{v}}_1. \quad & (\underline{\mathbf{\Theta}}_1(\underline{\mathbf{K}} + \underline{\mathbf{M}}) - \underline{\mathbf{Q}}\underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Lambda}}_1 + \underline{\mathbf{\Lambda}}_2 \\
& +\underline{\mathbf{\Lambda}}_2(\underline{\mathbf{\Theta}}_1 - \underline{\mathbf{\Theta}}_2)(\underline{\mathbf{I}} + \underline{\mathbf{R}} + \underline{\mathbf{\Phi}} - (\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}})\underline{\mathbf{\Theta}}_2 + \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_2^2)) \\
+\overrightarrow{\mathbf{v}}_2. \quad & (\underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}})) - \underline{\mathbf{M}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{R}\mathbf{\Theta}}_1) - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}}) \\
& -\underline{\mathbf{\Lambda}}_1(\underline{\mathbf{I}} + \underline{\mathbf{R}} + \underline{\mathbf{\Phi}}) + \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{\Phi}}(\underline{\mathbf{\Theta}}_2 - \underline{\mathbf{\Theta}}_1 - \underline{\mathbf{I}}) - \underline{\mathbf{I}} \\
& -\underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}}) + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}_1 + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}})\underline{\mathbf{\Theta}}_2 - \underline{\mathbf{R}\mathbf{\Phi}\mathbf{\Theta}}_2^2)) \\
+\overrightarrow{\mathbf{v}}_3. \quad & (\underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}}) + \underline{\mathbf{M}}(\underline{\mathbf{I}} + \underline{\mathbf{R}} + \underline{\mathbf{R}\mathbf{\Phi}}) - \underline{\mathbf{Q}}(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Theta}}_1\underline{\mathbf{\Phi}}) \\
& +\underline{\mathbf{\Lambda}}_1(\underline{\mathbf{R}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}\mathbf{\Phi}}) + \underline{\mathbf{\Lambda}}_2(\underline{\mathbf{\Phi}} + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{\Theta}}_1\underline{\mathbf{\Phi}} - \underline{\mathbf{\Phi}\mathbf{\Theta}}_2))) \\
+\overrightarrow{\mathbf{v}}_4. \quad & (\underline{\mathbf{Q}\mathbf{\Phi}\mathbf{R}} - \underline{\mathbf{\Phi}}(\underline{\mathbf{K}} + \underline{\mathbf{\Lambda}}_1\underline{\mathbf{R}} + \underline{\mathbf{\Lambda}}_2\underline{\mathbf{R}}) - \underline{\mathbf{M}\mathbf{R}}) = \overrightarrow{\mathbf{0}}
\end{aligned}$$

As with the finite queue, these equations are transformed to provide the boundary conditions for our infinite queue at the indicated *folded* levels:

Level 0

$$\overrightarrow{\mathbf{w}}_0. \begin{pmatrix} \underline{\mathbf{Q}}_0^0 & \underline{\mathbf{Q}}_0^1 \\ \underline{\mathbf{Q}}_1^0 & \underline{\mathbf{Q}}_1^1 \end{pmatrix} + \overrightarrow{\mathbf{w}}_1. \begin{pmatrix} \underline{\mathbf{Q}}_2^0 & \underline{\mathbf{Q}}_2^1 \\ \underline{\mathbf{0}} & \underline{\mathbf{Q}}_3^1 \end{pmatrix} = \overrightarrow{\mathbf{0}}$$

Level 1

$$\overrightarrow{\mathbf{w}}_0. \begin{pmatrix} \underline{\mathbf{Q}}_0^2 \\ \underline{\mathbf{Q}}_1^2 \end{pmatrix} + \overrightarrow{\mathbf{w}}_1. \begin{pmatrix} \underline{\mathbf{Q}}_2^2 \\ \underline{\mathbf{Q}}_3^2 \end{pmatrix} + \overrightarrow{\mathbf{w}}_1. \begin{pmatrix} \underline{\mathbf{Q}}_4^2 \\ \underline{\mathbf{0}} \end{pmatrix} = \overrightarrow{\mathbf{0}}$$

The first few probabilities in the folded queue are shown in figure 3.11. When unfolded, the distribution agrees with that given by the spectral expansion method, the first few levels being shown in figure 3.12.

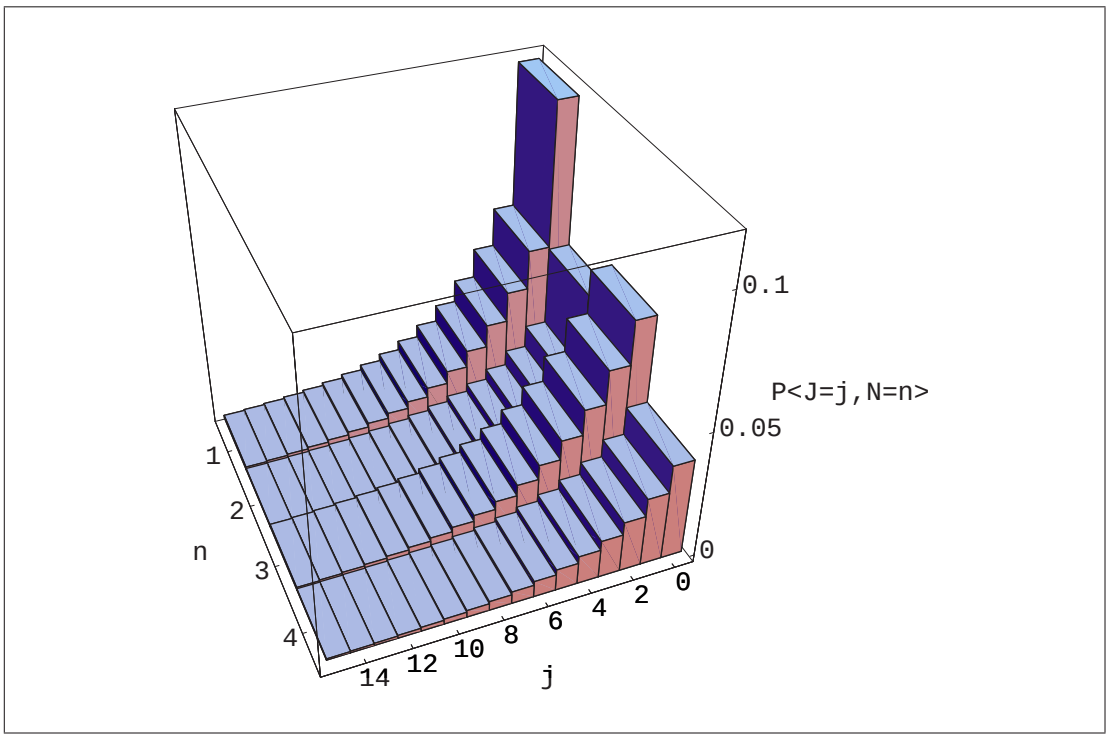


Figure 3.11: Infinite queue example: steady-state probability distribution of the first 16 *folded* levels.

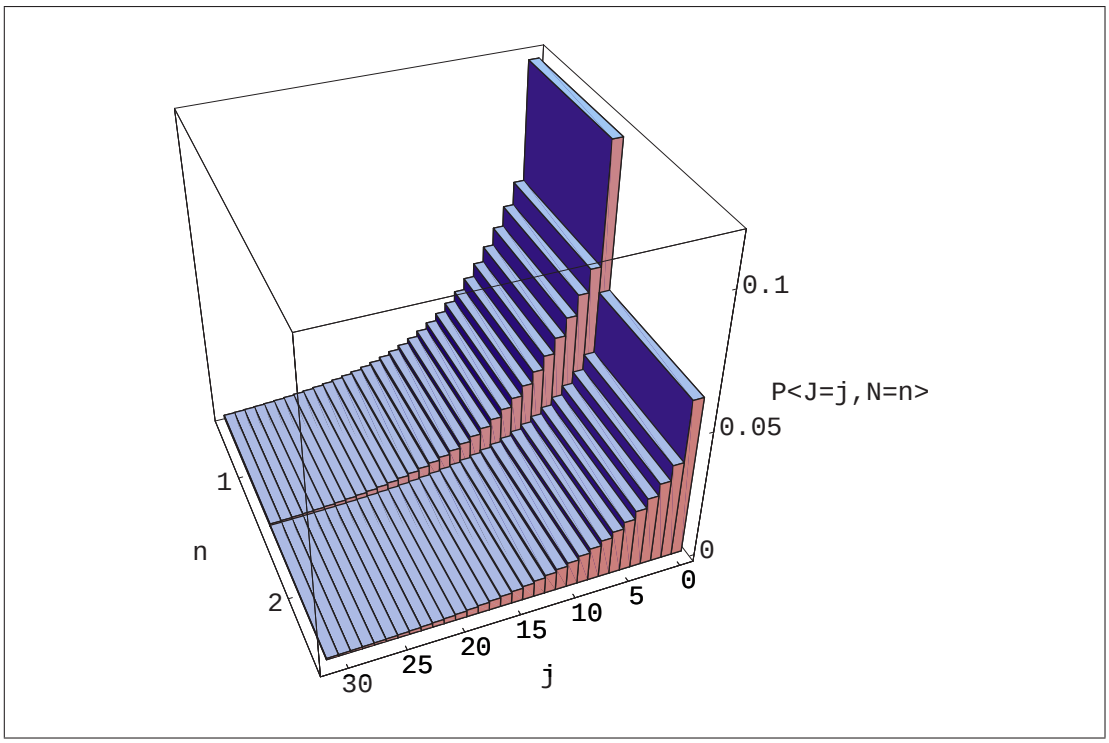


Figure 3.12: Infinite queue example: steady-state probability distribution of the first 32 levels obtained by spectral expansion or by *unfolding* the matrix geometric solution.

3.9 Accuracy

For simplicity of reproduction of results, the finite examples with two modulation states we used have odd maximum queue lengths to fit exactly into the folded structure. Solution to queues with even maximum queue lengths requires the inclusion of notional dummy states in the lattice with zero occupation probability, and is not additionally enlightening.

We use Mathematica® [Wol99] to support our analysis work, and results in this thesis were calculated using \$MinPrecision=20 (*i.e.* a precision of 10^{-20}). We used the built-in function `LUDecomposition` to solve the linear systems we described as matrix equations. Eigenvalues are found using the Mathematica function `Solve` applied to the polynomial determinant equation, and the eigenvectors by using `NullSpace` applied to the transpose of the characteristic matrix with the appropriate eigenvalue substituted.

For the finite waiting room example, we solved for the probability of queue states 1) directly using untransformed balance equations, 2) directly using localised balance equations, 3) directly using a folded system, 4) by spectral expansion of the localised equations, 5) by spectral expansion of the folded localised equations and 6) by matrix geometric methods applied to the folded localised equations. In every case, the results for the equilibrium occupation probabilities of the joint modulation and queue length states solved using methods 1, 2, 3, 4 and 5 were identical to within the numerical precision used. We therefore compare only spectral expansion and (unfolded) matrix geometric methods explicitly.

In solving the finite queues, we found that the matrix geometric methods returned results which differed from spectral expansion to a significant degree, of the order of $10^{-7} \ldots 10^{-5}$, *i.e.* not within the minimal precision. To illustrate this with meaningful values, we show the probability of the queue being empty and full for each modulation state, thus indicating the utilisation and blocking probability. The values are given for spectral expansion (SE) which agree with the direct solution of the explicit Markov chain, and matrix geometric methods (MG), which do not.

L	j	SE	MG
49	0	$(0.14265372, 0.07371261)$	$(0.1426\mathbf{4900}, 0.0737\mathbf{1004})$
	49	$(2.91845e-5, 1.11857e-5)$	$(2.9\mathbf{5770}e-5, 1.\mathbf{2028}e-5)$
15	0	$(0.1509138, 0.7798059)$	$(0.150\mathbf{5570}, 0.77\mathbf{74897})$
	15	$(1.0946906e-2, 0.4447833e-2)$	$(1.\mathbf{113158}e-2, 0.\mathbf{54306}e-2)$

3.9.1 The effect of raising λ to a power

The solution to the system using spectral expansion is

$$\overrightarrow{\mathbf{v}}_{j+\epsilon^b} = \sum_k \alpha_k \lambda_k^j \overrightarrow{\boldsymbol{\psi}}_k$$

but we use estimates $\hat{\lambda}_k$ and $\overrightarrow{\hat{\boldsymbol{\psi}}}$, which are perturbed by the numerical representation. Using the vector $\overrightarrow{\epsilon}$ to represent the maximum error in eigenvectors and ϵ_k for the error in the eigenvalues, we have an approximation $\overrightarrow{\hat{\mathbf{v}}}_j$ to the state occupation vectors:

$$\overrightarrow{\hat{\mathbf{v}}}_{j+\epsilon^b} = \sum_k \alpha_k (\lambda_k + \varepsilon_k)^j (\overrightarrow{\boldsymbol{\psi}}_k + \overrightarrow{\varepsilon})$$

This gives an error varying with j arising from

$$\hat{\lambda}^j - \lambda^j = (\lambda + \varepsilon)^j - \lambda^j = j\varepsilon\lambda^{j-1} + o(\varepsilon^2) \tag{3.35}$$

The term $j\varepsilon\lambda^{j-1}$ decreases with j when $j/(j+1) > \lambda$, which requires $\lambda < 1$ and j sufficiently large. If $\lambda > 1$, then the error always increases with j. The implications of this for infinite queues, in which eigenvalues on or outside the unit disc are excluded, is that errors are not amplified. In finite queues, however, in which all eigenvalues are included, errors in the larger eigenvalues will be amplified in the balance equations at the full queue. Note, however, that errors in the eigenvectors are not amplified. Thus, satisfaction of the boundary conditions at the full queue are not compromised, as these cover a small range of queue lengths differing by a small power of the eigenvalues.

3.9.2 The effect of raising F and B to powers

In the methods we provide the matrices providing the solution in the matrix geometric method encapsulate (between them) the entire Eigensystem of the solution. Either

or both of these may have eigenvalues greater than 1. Since the matrix $\underline{\mathbf{F}}$ provides components of the solution to both finite and infinite queues, and the approximation and error behaviour of $\underline{\hat{\mathbf{B}}}$ exhibits the same form of behaviour, we examine $\underline{\mathbf{F}}$ and its approximation in detail.

The numerical estimate $\underline{\hat{\mathbf{F}}} = \underline{\mathbf{F}} + \underline{\mathbf{D}}$, where $\underline{\mathbf{F}}$ is the "perfect" solution, and $\underline{\mathbf{D}}$ is a matrix of errors due principally to rounding in the numerical representation. Examining the role of the term $\underline{\mathbf{D}}$ in the solution, we find a term which does not simplify in the same way as in expression (3.35), as the matrices are not commutative: expanding the result of raising the numerical estimate $\underline{\hat{\mathbf{F}}}$ to the power j yields the following:

$$\begin{aligned}
\underline{\hat{\mathbf{F}}}^j &= (\underline{\mathbf{F}} + \underline{\mathbf{D}})^j \\
&= \underline{\mathbf{F}}^j + (\underline{\mathbf{D}}\underline{\mathbf{F}}^{j-1} + \underline{\mathbf{F}}\underline{\mathbf{D}}\underline{\mathbf{F}}^{j-2} + \cdots + \underline{\mathbf{F}}^{j-1}\underline{\mathbf{D}}) + o(\underline{\mathbf{D}}^2) \\
&= \underline{\mathbf{F}}^j + (\underline{\mathbf{\Omega}}_0 + \underline{\mathbf{\Omega}}_1 + \cdots + \underline{\mathbf{\Omega}}_{j-1}) + o(\underline{\mathbf{D}}^2) \\
&= \underline{\mathbf{F}}^j + \underline{\mathbf{\Omega}} + o(\underline{\mathbf{D}}^2)
\end{aligned}$$

where $\underline{\mathbf{\Omega}}_i = \underline{\mathbf{F}}^i\underline{\mathbf{D}}\underline{\mathbf{F}}^{j-1-i}$ and $\underline{\mathbf{\Omega}} = \sum_i \underline{\mathbf{\Omega}}_i$. To obtain an error estimate, we consider the effect that the perturbation $\underline{\mathbf{D}}$ has on the eigenvectors of $\underline{\hat{\mathbf{F}}}$. Let matrix $\underline{\mathbf{F}}$ have eigenvectors $\overrightarrow{\boldsymbol{\psi}}_i$ associated with eigenvalues λ_i labelled such that $\lambda_i \leq \lambda_{i+1}$.

As $\underline{\mathbf{D}}$ is a perturbation matrix with small entries distributed about zero, we have that (in general) for any $1 \leq i \leq N$,

$$\overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{D}} = \sum_{k=1}^{N} \beta_{i,k}\overrightarrow{\boldsymbol{\psi}}_k$$

where $\beta_{i,k}$ are also small.

Consider the h^{th} error term, $\underline{\mathbf{\Omega}}_h = \underline{\mathbf{F}}^h\underline{\mathbf{D}}\underline{\mathbf{F}}^{j-1-h}$, when eigenvector $\overrightarrow{\boldsymbol{\psi}}_i$ is multiplied by it

$$\begin{aligned}
\overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{\Omega}}_h &= \overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{F}}^h\underline{\mathbf{D}}\underline{\mathbf{F}}^{j-1-h} \\
&= \lambda_i^h\overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{D}}\underline{\mathbf{F}}^{j-1-h} \\
&= \lambda_i^h\left(\textstyle\sum_{k=1}^{N} \beta_{i,k}\overrightarrow{\boldsymbol{\psi}}_k\right)\underline{\mathbf{F}}^{j-1-h} \\
&= \lambda_i^h\textstyle\sum_{k=1}^{N} \beta_{i,k}\lambda_k^{j-1-h}\overrightarrow{\boldsymbol{\psi}}_k
\end{aligned}$$

to evaluate the total error in evaluating $\overrightarrow{\boldsymbol{\psi}}_i\hat{\underline{\mathbf{F}}}^j$, we sum up all the individual errors

$$\begin{aligned}
\overrightarrow{\boldsymbol{\psi}}_i\underline{\boldsymbol{\Omega}} &= \overrightarrow{\boldsymbol{\psi}}_i(\underline{\mathbf{DF}}^{j-1} + \underline{\mathbf{FDF}}^{j-2} + \cdots + \underline{\mathbf{F}}^{j-1}\underline{\mathbf{D}}) \\
&= \textstyle\sum_{h=0}^{j-1} \overrightarrow{\boldsymbol{\psi}}_i\underline{\boldsymbol{\Omega}}_h \\
&= \textstyle\sum_{h=0}^{j-1} \left(\lambda_i^h \sum_{k=1}^{N} \beta_{i,k}\lambda_k^{j-1-h}\overrightarrow{\boldsymbol{\psi}}_k\right) \\
&= \textstyle\sum_{k=1}^{N} \left(\beta_{i,k}\overrightarrow{\boldsymbol{\psi}}_k \sum_{h=0}^{j-1} \lambda_i^h\lambda_k^{j-1-h}\right) \\
&= \textstyle\sum_{k=1}^{N} \beta_{i,k}\overrightarrow{\boldsymbol{\psi}}_k \frac{\lambda_i^j-\lambda_k^j}{\lambda_i-\lambda_k}
\end{aligned}$$

The behaviour for large j is prescribed by the largest eigenvalue, λ_N, as

$$\overrightarrow{\boldsymbol{\psi}}_i\underline{\boldsymbol{\Omega}} \doteq \begin{cases} \beta_{i,N}\overrightarrow{\boldsymbol{\psi}}_N\lambda_N^{j-1} & \text{if } i \neq N \\ \sum_{k=1}^{N-1} \beta_{i,k}\overrightarrow{\boldsymbol{\psi}}_k\lambda_N^{j-1} & \text{if } i = N \end{cases}$$

We can now quantify the asymptotic effect of numerical inaccuracy in $\hat{\underline{\mathbf{F}}}$ on the eigenvectors of $\underline{\mathbf{F}}$.

$$\begin{aligned}
\overrightarrow{\boldsymbol{\psi}}_i\hat{\underline{\mathbf{F}}}^j &= \overrightarrow{\boldsymbol{\psi}}_i(\underline{\mathbf{F}}^j + \underline{\boldsymbol{\Omega}} + o(\underline{\mathbf{D}}^2)) \\
&\doteq \lambda_i^j\overrightarrow{\boldsymbol{\psi}}_i + \begin{cases} \beta_{i,N}\overrightarrow{\boldsymbol{\psi}}_N\lambda_N^{j-1} & \text{if } i \neq N \\ \sum_{k=1}^{N-1} \beta_{i,k}\overrightarrow{\boldsymbol{\psi}}_k\lambda_N^{j-1} & \text{if } i = N \end{cases} \\
&\doteq \begin{cases} \beta_{i,N}\overrightarrow{\boldsymbol{\psi}}_N\lambda_N^{j-1} & \text{if } i \neq N \\ \lambda_N^j\overrightarrow{\boldsymbol{\psi}}_N & \text{if } i = N \end{cases}
\end{aligned} \tag{3.36}$$

which means that every eigenvector $\overrightarrow{\boldsymbol{\psi}}_i$ asymptotically approaches the eigenvector $\overrightarrow{\boldsymbol{\psi}}_N$ when multiplied by large powers of $\hat{\underline{\mathbf{F}}}$.

This behaviour is mirrored in the use of $\hat{\underline{\mathbf{B}}}$ in providing the additional Eigensystem components required for use in finite queues.

This tendency for any vector, when multiplied by high powers of the matrix geometric matrix $\hat{\underline{\mathbf{F}}}$ to approach the eigenvector associated with the largest eigenvalue is not surprising, as this phenomenon gives us the well-known power method for finding the largest eigenvalue of a matrix.

Perturbation error in infinite queues

The above error estimate shows that the numerical solution for large j becomes dominated by the eigenvector with largest eigenvalue. Within the solution process for

infinite systems all eigenvalues – including λ_N – are contained strictly within the unit disc due to the requirement of normalisation. Consequently, for all i, the error estimate $\overrightarrow{\boldsymbol{\psi}}_i\underline{\boldsymbol{\Omega}}$ dominates the probability solution vector for large j, but also vanishes, allowing an accurate solution to be achieved using the matrix geometric method.

Perturbation error in finite queues

In large, finite systems, the largest eigenvalue λ_N is outside the unit disc and the error term $\overrightarrow{\boldsymbol{\psi}}_i\underline{\boldsymbol{\Omega}}$ ceases to vanish so that, with increasing queue length, numerical errors are amplified. The particular value of $\mathbf{f}\underline{\mathbf{F}}^{j-\epsilon^b}$ (from $\overrightarrow{\mathbf{v}}_j = \overrightarrow{\mathbf{v}}_{\epsilon^b}\underline{\mathbf{F}}^{j-\epsilon^b} + \overrightarrow{\mathbf{v}}_{\epsilon^t}\underline{\mathbf{B}}^{\epsilon^t-j}$) may not be particularly inaccurate, because in the perfect solution, the largest eigenvalue will dominate. However, the contributions from the other components to the solution, in particular those arising from the smaller eigenvalues, quickly lose precision. As a consequence the boundary condition equations that specify $\overrightarrow{\mathbf{f}}$ and $\overrightarrow{\mathbf{b}}$, linking the top and bottom of the queue, can only be evaluated in an inaccurate manner. Subsequent solution yields similarly inaccurate values for $\overrightarrow{\mathbf{f}}$ and $\overrightarrow{\mathbf{b}}$, which cause inaccuracies throughout the queue.

3.9.3 Numerical example of $\mathbf{f}.\hat{F}^j$

We have presented an asymptotic error estimate for calculations performed in solving a finite queue using the matrix geometric method. The limit $j \to \infty$ is never achieved for any given problem so we show the behaviour of the eigenvectors in our example to illustrate how quickly errors can accumulate. To this end we examine the angle $\alpha(\overrightarrow{\boldsymbol{\psi}}_i\hat{\underline{\mathbf{F}}^j}, \overrightarrow{\boldsymbol{\psi}}_4) = \cos^{-1}\frac{\overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{F}}^j.\overrightarrow{\boldsymbol{\psi}}_4}{\overrightarrow{\boldsymbol{\psi}}_i\underline{\mathbf{F}}^j \quad \overrightarrow{\boldsymbol{\psi}}_4}$ ($\overrightarrow{\boldsymbol{\psi}}_4$ being the dominant eigenvector and $\left\|\overrightarrow{\mathbf{x}}\right\|_2$ the Euclidean norm of a vector $\overrightarrow{\mathbf{x}}$) for various j when working with double precision conforming with IEEE floating point standards. In figure 3.13, we show the convergence of each of the non-dominant eigenvectors toward the *direction*[11] of the dominant eigenvector applied to successive powers of $\hat{\underline{\mathbf{F}}}$. The vertical axis between 0 and 1 indi-

[11]The relative magnitude of the vectors was not considered

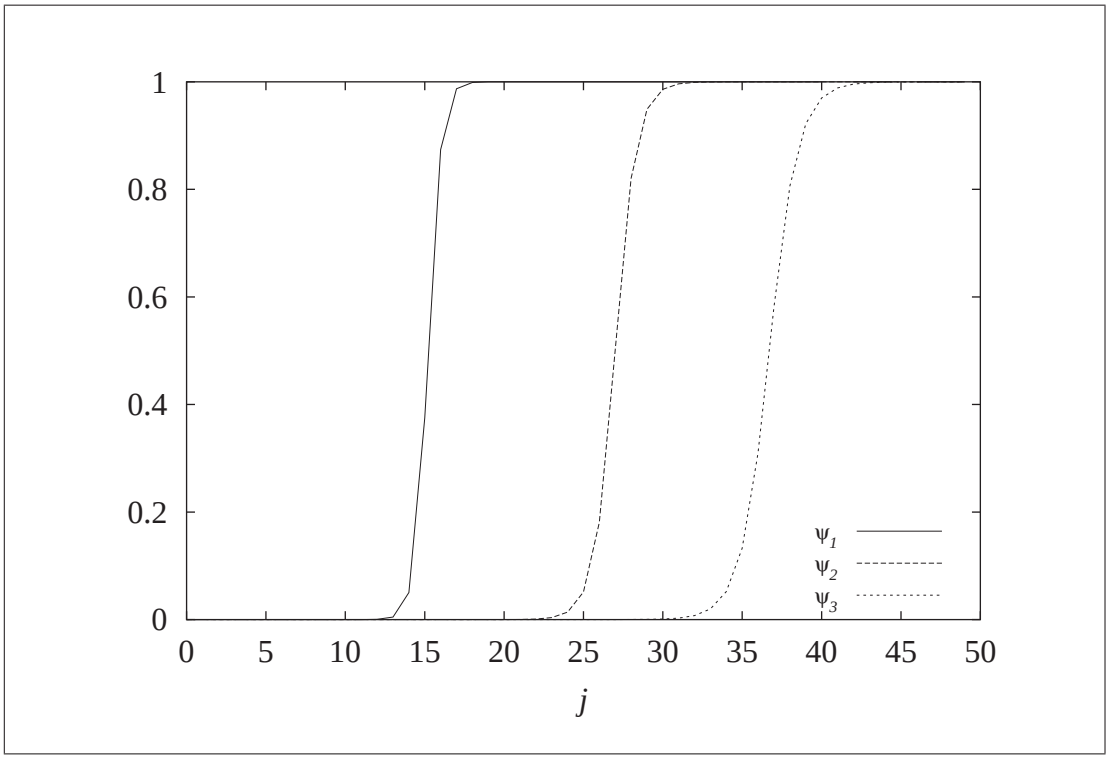


Figure 3.13: Progress toward convergence with the dominant eigenvector with power j of matrix F.

cates the progression in terms of the fraction of the angle (given by its cosine) between the true vector and the dominant vector traversed.

We might wish to reduce this effect by considering alternative schemes for constructing the value of $\mathbf{f}.\hat{\underline{\mathbf{F}}}^{j}$. The graph above arises from successive post-multiplication by $\underline{\mathbf{F}}$ of an intermediate result, seeded at $\mathbf{f}$. If instead, we attempt to calculate $\underline{\mathbf{F}}^{j}$ more efficiently by use of intermediate, previously calculated squares of the matrix, *e.g.* we use $\hat{\underline{\mathbf{F}}}^{4} = \hat{\underline{\mathbf{F}}}^{2}\hat{\underline{\mathbf{F}}}^{2}$ instead of $\hat{\underline{\mathbf{F}}}^{4} = \underline{\hat{\mathbf{F}}\hat{\mathbf{F}}\hat{\mathbf{F}}\hat{\mathbf{F}}}$, the trajectory of the estimate of $\mathbf{f}.\hat{\underline{\mathbf{F}}}^{j}$ becomes erratic after the initial departure from it's correct direction, but gives overall smaller errors.

No matter how accurate the matrix $\hat{\underline{\mathbf{F}}}^{j}$ is calculated, the lower error bound (3.36) still holds as its derivation only assumed an initial error $\underline{\mathbf{D}}$ in the matrix and exclusively used symbolic manipulations the remainder of calculations.

3.9.4 Implementational issues

The derivation of the $\underline{\hat{\mathbf{F}}}$ and $\underline{\hat{\mathbf{B}}}$ matrices used in the matrix geometric method is most easily obtained using the Latouche-Ramaswami (LR) method. Our localisation and folding procedures ensure that the recurrence relation is of the required form for LR to successfully solve. More modern derivatives like cyclic reduction [BinMei97] exist, but the already excellent convergence properties of LR make this choice entirely adequate.

For the spectral expansion method, we explicitly require the eigenvalues and eigenvectors within the repeating region represented by our recurrence relation. Eigenvalues are either found by using dedicated Eigenroutines, by direct solving for the roots of the characteristic polynomial or a combination of the two. Particular numerical difficulties arise when eigenvalues occur multiply or are very close to each other. While it is possible to choose queueing parameters that give rise to pairs of eigenvalues arbitrarily close to each other (see section 3.4.1), such behaviour is untypical in practice. Most numerical packages which offer eigenvalue/eigenvector solving in use today are derived from [WilRei71]. A good, public-domain implementation in FORTRAN is given in [Smi+76], whereas for examples in the C language [PreTeu+92] is invaluable. Development and testing work was done by solving the characteristic polynomial $\det|\underline{\mathbf{A}} - \lambda I| = 0$ using Mathematica [Wol99], which was very reliable in finding all eigenvalues even for relatively large problems where more than 200 are present. For infinite queues, where only the eigenvalues $0 < |\lambda_i| < 1$ are of interest, a simple bracketing algorithm [PreTeu+92] on this range can be used. Once the approximate locations of the roots are known, we use root polishing [PreTeu+92] to gain high precision. Finite queues are initially treated the same. Once the eigenvalues within the unit circle are known, we deflate [PreTeu+92] the characteristic polynomial and proceed with the solving for the other roots.

As an alternative to solving the characteristic equation, spectral expansion can borrow from the efficient LR algorithm used in the MG method. For a finite queue, it provides two matrices that, collectively, encapsulate all eigenvalues present. It is therefore possible to apply separate eigenvalue analysis to both matrices, which offers a

considerable computational advantage over direct analysis of the full matrix $\underline{\mathbf{A}}$ given in 3.7 that gives a higher order system. Once approximate eigenvalues are found, root polishing [PreTeu+92] can be performed. Eigenvectors are subsequently found using standard techniques [PreTeu+92]. In this manner we combine the greater efficiency of the LR method with the improved accuracy and numerical stability offered by the spectral expansion representation. Additionally, the spectral expansion representation is required for the calculation of sojourn time in the next chapter, as the distributions are given in terms of eigenvalues and eigenvectors which would otherwise not be available.

Chapter 4

Response time of single queue

4.1 Introduction

In this chapter we use the steady-state results to obtain sojourn (or response) time distributions for the types in chapter 2. All but one queue feature is covered: if there are to be negative customers, they need to be of the RCH (head-per killing) variety. When they are not, *i.e.* customers are removed from the tail of the queue (as in tail-safe or tail-vulnerable) we can derive expressions in the Laplace domain (*e.g.* [Har02]), but unlike the RCH case we present here, these have not yet been successfully inverted in an analytic manner.

The method we use to find sojourn times relies on both Laplace transforms and generating functions. As illustrative example, we first consider a simple, infinite single-server G-queue with no batches or modulation. The queue experiences Poisson arrivals with rate λ, exponential service times with parameter μ and RCH (h^p killing) negative customer arrivals with rate κ. It is easily verifiable that the probability distribution of this queue at steady-state is given by

$$\pi_j = (1-\rho)\rho^j \quad \text{where} \quad \rho = \tfrac{\lambda}{\mu+\kappa}$$

To derive a response time distribution for this queue, we consider the progress of a special *tagged* customer through our queue. The random variable $A(x)$ is used to denote the number of customers ahead of it at time x and T is the time remaining,

without loss of generality at time 0, up to its departure. For $j \geq 0$, we define the probability distribution of T jointly with the probability of the tagged customer not being killed, conditional on j customers ahead of the tagged customer initially, as $F_j(t) = P(T \leq t | A(0) = j)$. Thus $F_j(\infty) = \mu/(\mu + \kappa)$.

Using the Markov property and stationarity, we consider a small initial interval h to give an expression for $F_j(t + h)$ in terms of $\{F_a(t) | a \geq 0\}$.

We have

$$F_0(t + h) = F_0(t)(1 - \mu h - \kappa h) + \mu h$$

and

$$F_j(t + h) = F_j(t)(1 - \mu h - \kappa h) + F_{j-1}(t)(\mu h + \kappa h)$$

for $j \geq 1$.

By taking the appropriate terms to the left hand side and dividing by h, the following differential-difference equations are derived:

$$\begin{aligned} \tfrac{dF_0(t)}{dt} &= \mu - F_0(t)(\mu + \kappa) \\ \tfrac{dF_j(t)}{dt} &= F_{j-1}(t)(\mu + \kappa) - F_j(t)(\mu + \kappa) \end{aligned}$$

In principle, it is now possible to solve these first order differential equations using the integrating factor method to find $F_j(t)$, but we use an alternative method, relying on Laplace transforms, instead. The rationale for using this method is that it is much easier to generalise, as work in [Har02] demonstrates.

We define the Laplace transform of the distribution functions $F_j(t)$ as

$$L_j(s) = \int_0^\infty e^{-st} F_j(t) dt$$

Applying the Laplace transform to our differential-difference equations, we get

$$\begin{aligned} sL_0 &= \tfrac{\mu}{s} - (\mu + \kappa)L_0 \\ sL_j &= (\mu + \kappa)L_{j-1} - (\mu + \kappa)L_j \end{aligned}$$

or, equivalently

$$\begin{aligned} L_0 &= \tfrac{\mu}{s(s+\mu+\kappa)} \\ (s + \mu + \kappa)L_j &= (\mu + \kappa)L_{j-1} \end{aligned} \tag{4.1}$$

This system of equations can be easily solved using direct methods. We will be using generating functions instead, which – while not necessary here – are again easier to generalise for more complex queues.

Using the definition $D(z,s) = \sum_{j=1}^{\infty} L_j(s) z^j$, we multiply the second equation in (4.1) by z^j and sum from $j = 1$ to ∞, so that

$$\begin{aligned}
(s+\mu+\kappa)\sum_{j=1}^{\infty} L_j(s) z^j &= (\mu+\kappa)\sum_{j=1}^{\infty} L_{j-1} z^j \\
(s+\mu+\kappa)D(z,s) &= (\mu+\kappa)\sum_{j=0}^{\infty} L_j z^{j+1} \\
(s+\mu+\kappa)D(z,s) &= (\mu+\kappa)(zL_0 + z\sum_{j=1}^{\infty} L_j z^j) \\
(s+\mu+\kappa)D(z,s) &= z(\mu+\kappa)L_0 + z(\mu+\kappa)D(z,s) \\
D(z,s) &= \frac{z(\mu+\kappa)}{s+(\mu+\kappa)(1-z)} L_0
\end{aligned}$$

Substituting the value for L_0 in (4.1),

$$D(z,s) = \frac{z(\mu+\kappa)\mu}{s(s+\mu+\kappa)(\mu+\kappa+s-(\mu+\kappa)z)}$$

Using the generating function, we now express the Laplace transform $L(s)$ of the desired response time $F(t)$ as follows

$$\begin{aligned}
L(s) &= \textstyle\sum_{j=0}^{\infty} \pi_j L_j(s) \\
&= \pi_0 L_0(s) + \textstyle\sum_{j=1}^{\infty} \pi_j L_j(s) \\
&= (1-\rho)\rho^0 L_0(s) + \textstyle\sum_{j=1}^{\infty} (1-\rho)\rho^j L_j(s) \\
&= (1-\rho)L_0(s) + (1-\rho)D(\rho,s) \\
&= \tfrac{\mu(\mu+\kappa-\lambda)}{s(\mu+\kappa)(s+\mu+\kappa+-\lambda)} \\
&= \tfrac{\mu}{\mu+\kappa}\left(\tfrac{1}{s} + \tfrac{1}{s+\mu+\kappa-\lambda}\right)
\end{aligned}$$

This Laplace transform can be inverted symbolically, resulting in the expected result

$$F(t) = \frac{\mu}{\mu+\kappa}\left(1 - e^{-t(\mu+\kappa-\lambda)}\right)$$

4.2 A modulated multi-server with batch transitions

To generalise these results, we use expressions presented in [HarZat04], where the Laplace transform of the response time density of an infinite modulated queue, with one positive customer stream, one service stream and one negative customer stream, is given. The derivation itself is done in a similar manner to that used for the $M/M/1$-queue in that generating functions are used to describe the Laplace densities. Here, we show that the resulting Laplace transforms are always algebraically invertible and we develop more general, automated formulations incorporating multiple streams of all types, so that the response time results can be used in a more general network setting where queues have more than one arrival stream.

We seek the probability distributions

$$\overrightarrow{\mathbf{F}}_j(t) = (F_{1j}(t), \ldots, F_{Nj}(t))$$

where, for $1 \leq k \leq N$ and $j \geq 0$,

$$F_{kj} = P\left(T \leq t | I(0) = k, A(0) = j\right)$$

The random variable $I(x)$ is the phase in which the tagged customer finds the modulation chain of the queue at time x. As in section 4.1, $A(x)$ denotes the number of customers ahead of the tagged customer at time x, and T is the time remaining, without loss of generality at time 0, up to its departure.

In [Har02, HarZat04], using the same steps as in the simple $M/M/1$ G-queue, an expression is derived for the unconditional Laplace transform of the sojourn time density. In particular, it is found that within the processing region, $0 \leq j \leq c-1$, the sojourn times are identically distributed.

$$\overrightarrow{\mathbf{L}}_0(s) = \overrightarrow{\mathbf{L}}_1(s) = \ldots = \overrightarrow{\mathbf{L}}_{c-1}(s) = (s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c)^{-1}\underline{\mathbf{M}}\,\overrightarrow{\mathbf{e}}/s \qquad (4.2)$$

For the repeating region (*c.f.* section 3.2) a generating function $\overrightarrow{\mathbf{D}}(z,s) = \sum_{j=c}^{\infty} \overrightarrow{\mathbf{L}}_j z^j$ is used, whose value is determined by the equation

$$\overrightarrow{\mathbf{D}}(z,s) = \underline{\mathbf{C}}^{-1}(z,s)\,\overrightarrow{\mathbf{b}}(z,s)$$

where

$$\begin{aligned}\underline{\mathbf{C}}(z,s) &= s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{K}} + c\underline{\mathbf{M}} - \underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R}}z)^{-1}z \\ &- c\underline{\mathbf{M}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}z)^{-1}z\end{aligned} \tag{4.3}$$

$$\begin{aligned}\overrightarrow{\mathbf{b}}(z,s) &= (cz^c/s)\underline{\mathbf{M}\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}z)^{-1}\overrightarrow{\mathbf{e}} \\ &+ z^c\left(\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R}}z)^{-1} + c\underline{\mathbf{M}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}z)^{-1}\right)\overrightarrow{\mathbf{L}}_0(s)\end{aligned} \tag{4.4}$$

Once $\overrightarrow{\mathbf{L}}_0(s)$ and $\overrightarrow{\mathbf{D}}(z,s)$ are found, the Laplace transform of the unconditional sojourn time density function can be written as follows (*c.f.* [Har02, HarZat04])

$$L(s) = \frac{1}{\lambda^*}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\overrightarrow{\mathbf{L}}_0(s)\sum_{j=0}^{c-1}\left(\sum_{q=0}^{j}\underline{\mathbf{\Theta}}^{j-q}\underline{\mathbf{\Lambda}}\overrightarrow{\mathbf{v}}_q\right) \tag{4.5}$$

$$-\frac{1}{\lambda^*}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}^{-1}\sum_{k=1}^{N}a_k\xi_k(\underline{\mathbf{I}} - \xi_k\underline{\mathbf{\Theta}}^{-1})^{-1}\underline{\mathbf{\Lambda}}\overrightarrow{\psi}_k\overrightarrow{\mathbf{D}}(\xi_k,s)$$

Here, ξ_k and $\overrightarrow{\psi}_k$ are the k^{th} eigenvalues and eigenvectors used in the spectral expansion method to find the steady-state state occupation probability distribution. There are N of these, because we are working with an infinite queue with one arrival stream (see section 3.4.1). $\overrightarrow{\mathbf{v}}_q$ is the state occupation probability of level q and $\lambda^* = \sum_{h=1}^{N}\pi_h\lambda_h$ is the average arrival rate to the queue, where π_h is the steady-state probability of phase h in the modulation process, as introduced in section 2.2.

In the absence of arrival batches, that is when $\underline{\mathbf{\Theta}}$ is the zero-matrix (and therefore cannot be inverted), the following formula takes the place of (4.5).

$$L(s) = \frac{1}{\lambda^*}\overrightarrow{\mathbf{L}}_0\sum_{j=0}^{c-1}\overrightarrow{\mathbf{v}}_j + \frac{1}{\lambda^*}\sum_{k=1}^{N}a_k\overrightarrow{\mathbf{D}}(\xi_k,s)\overrightarrow{\psi}_k \tag{4.6}$$

4.3 Inverting the Laplace transform

For practical purposes, the expression (4.5) needs to be inverted to yield the function $F(t)$, the sojourn time distribution in the time domain. Traditionally, this was either deemed too difficult and not done at all, or was carried out by way of expensive numerical inversion techniques (*e.g.* [HarKno02]). By proving a series of lemmas, we show

that the Laplace expression (4.5) is always algebraically invertible, so that we have the exact sojourn time densities at our disposal, without need for numerical inversion.

For all z, the matrix $\underline{\mathbf{C}}(z,s)$ given by equation (4.3) is of the form $\underline{\mathbf{C}}(z,s) = \underline{\mathbf{\Delta}}(z) - \underline{\mathbf{Q}} + s\underline{\mathbf{I}}$, where $\underline{\mathbf{\Delta}}(z) = \underline{\mathbf{diag}}(d_1(z), \ldots, d_N(z))$ is a diagonal matrix with every diagonal element $d_i(z) > 0$.

Lemma 1. *The real parts of the eigenvalues of the matrix $\underline{\mathbf{P}}(z) = \underline{\mathbf{\Delta}}(z) - \underline{\mathbf{Q}}$ are positive.*

Proof The off-diagonal elements of $\underline{\mathbf{Q}}$ are rates and therefore positive. The diagonal elements of $\underline{\mathbf{Q}}$ are the negative row sums of the off-diagonals.

$$(\underline{\mathbf{Q}})_{i,i} = -\sum_{j=1, j\neq i}^{N} q_{i,j}$$

Consequently, $\underline{\mathbf{P}}(z)$ is strictly diagonally dominant, with a positive diagonal. Using Gershgorin's circle theorem [Kre88] on $\underline{\mathbf{P}}(z)$ shows that the real parts of all eigenvalues are positive. ■

Let $\mathcal{P}^n(x)$ be the set of polynomials in the variable x of order n.

Lemma 2. *For all z, the elements of $\underline{\mathbf{C}}^{-1}(z,s)$ are rational functions in s of the form q/p, where $p \in \mathcal{P}^N(s)$ and $q \in \mathcal{P}^{N-1}(s)$. The roots of p depend (in general) on z and have a negative real part.*

Proof By definition of the inverse,

$$\underline{\mathbf{C}}^{-1}(z,s) = \frac{adj(\underline{\mathbf{C}}(z,s))}{\det(\underline{\mathbf{C}}(z,s))} = \frac{adj(\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}})}{\det(\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}})}$$

where the adjoint matrix, $adj(\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}})$, is the transpose of the matrix of cofactors of $\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}}$. The determinant of $\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}}$ is a polynomial in s of order N. All cofactors are polynomials in s of order $N-1$. Hence the elements of $\underline{\mathbf{C}}^{-1}(z,s)$ are of the form q/p, where $p \in \mathcal{P}^N(s)$ and $q \in \mathcal{P}^{N-1}(s)$.

For the second part of the lemma, we note that the roots of $\det(\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}})$ are the negatives of the eigenvalues of $\underline{\mathbf{P}}(z)$. The matrix $\underline{\mathbf{P}}(z)$ has been shown to have only eigenvalues with positive real parts. Consequently, the roots of $p(s) = \det(\underline{\mathbf{P}}(z) + s\underline{\mathbf{I}})$ depend on z and they have negative real parts as required. ■

Lemma 3. *The elements of $\overrightarrow{\mathbf{L}}_0(s)$ given in (4.2) are rational functions in s of the form $t/(sr)$, where $r \in \mathcal{P}^N(s)$ and $t \in \mathcal{P}^{N-1}(s)$. The roots of r have negative real parts and are independent of z.*

Proof From (4.2), $s\overrightarrow{\mathbf{L}}_0(s) = (s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c)^{-1}\underline{\mathbf{M}}\overrightarrow{\mathbf{e}}$ and so,

$$(s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c)^{-1} = \frac{adj(s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c)}{\det(s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c)}$$

The matrix $-\underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c$ is diagonally dominant with a positive diagonal and using the same argument as that in the previous lemma, we see that each component of $\overrightarrow{\mathbf{L}}_0$ takes the form $\overrightarrow{\mathbf{L}}_0.\overrightarrow{\mathbf{0}}e_i = t/sr$ where r is a polynomial whose roots have negative real parts. $r \in \mathcal{P}^N(s)$, $t \in \mathcal{P}^{N-1}(s)$ and the roots of r are obviously independent of z. ■

Lemma 4. *For all z, the elements of $\overrightarrow{\mathbf{b}}(z,s)$ given in (4.4) are rational functions in s of the form $t/(sr)$, $r \in \mathcal{P}^N(s)$ and $t \in \mathcal{P}^N(s)$. In addition, the roots of r are the same as those found for $\overrightarrow{\mathbf{L}}_0$ in the previous Lemma.*

Proof $\overrightarrow{\mathbf{b}}(z,s)$ is of the form $\overrightarrow{\mathbf{b}}(z,s) = \underline{\mathbf{\Delta}}_1(z)\overrightarrow{\mathbf{e}}_N/s + \underline{\mathbf{\Delta}}_2(z)\overrightarrow{\mathbf{L}}_0(s)$ where the $\underline{\mathbf{\Delta}}_1(z)$ and $\underline{\mathbf{\Delta}}_2$ are (in general) distinct diagonal matrices with non-zero values on their diagonal. The i^{th} component of $\overrightarrow{\mathbf{b}}(z,s)$ is therefore of the form

$$b_i = \frac{c_{i,1}}{s} + c_{i,2}L_{0,i} = \frac{c_{i,1}}{s} + \frac{c_{i,2}t}{sr} = \frac{c_{i,1}r + c_{i,2}t}{sr}$$

where r and t are as in the previous lemma and $c_{i,1}$ and $c_{i,2}$ are independent of s. ■

Lemma 5. *For all z, the elements of $\overrightarrow{\mathbf{D}}(z,s)$ are rational functions in s of the form $u/(sv)$, where $v \in \mathcal{P}^{2N}(s)$ and $u \in \mathcal{P}^{2N-1}(s)$. Of the $2N$ roots of v, exactly N are independent of z and N are dependent on z. In addition, for all z, all roots of v are either real and negative, or occur in complex conjugate pairs, with negative real parts.*

The proof follows from the previous three lemmas. ■

Lemma 6. *The Laplace transform $L(s)$ is analytically invertible and gives an unconditional sojourn time density function which is a mixture of exponential and Erlang densities as well as those of the types $e^{at}\cos(bt)$ and $e^{at}\sin(bt)$, where a is negative.*

Proof We consider the expression for $L(s)$ given in (4.5). It is a sum of individual terms, each of which depends on s through either the function $\overrightarrow{\mathbf{L}}_0(s)$ or $\overrightarrow{\mathbf{D}}(z,s)$, for some particular value of z. The elements of both of these functions have been shown to be rational functions of the form $u/(sv)$, where u and v are polynomials in s. v has real roots $\{x_1,\ldots,x_k\}$ and roots that form complex conjugate pairs $\{y_1 \pm iz_1,\ldots,y_l \pm iz_l\}$. The polynomial v is of higher order than u and it is therefore possible to use partial fractions to re-write each term as follows

$$\begin{aligned}\frac{u}{sv} &= \frac{c_0}{s} + \frac{c_1}{(s-x_1)^{m_1}} + \ldots + \frac{c_k}{(s-x_k)^{m_k}} \\ &+ \frac{d_1}{(s-(y_1+z_1))^{m_1}} + \ldots + \frac{d_l}{(s-(y_l+z_l))^{m_l}} \\ &+ \frac{e_1}{(s-(y_1-z_1))^{m_1}} + \ldots + \frac{e_l}{(s-(y_l-z_l))^{m_l}}\end{aligned} \tag{4.7}$$

Each of these terms can be algebraically inverted using the rule that the unique inverse of a Laplace transform $L(s) = \frac{1}{(s-\alpha)^n}$ is $F(t) = \frac{1}{(n-1)!}t^{n-1}e^{\alpha t}$. Hence, every term of $L(s)$ can be algebraically inverted. The real roots x_k are negative, so that their contribution to the resulting unconditional sojourn time $F(t)$ consists of Erlang-n[1] distributions $\frac{1}{(n-1)!}t^{n-1}e^{\alpha t}$, where α is non-positive.

The roots that occur in complex conjugate pairs can be inverted using the same formula as for the purely real ones. The two Laplace terms

$$\frac{d_k}{s-(a+ib)} \qquad\qquad \frac{e_k}{s-(a-ib)}$$

have as their unique inverse

$$d_k e^{(a+ib)t} = d_k e^{at}e^{ibt} = d_k e^{at}(\cos(bt) + i\sin(bt))$$

and

$$e_k e^{(a-ib)t} = e_k e^{at}e^{-ibt} = e_k e^{at}(\cos(bt) - i\sin(bt))$$

The sojourn time distribution must be real. This requires that imaginary parts that appear due to a complex conjugate pair of eigenvalues to cancel *for all values of t*. This is only guaranteed when the coefficients d_k and e_k also form a complex conjugate pair.

[1] Usually it is found that $n = 1$

Writing $d_k = a_k + ib_k$ and $e_k = a_k - ib_k$, we evaluate $d_k e^{(a+ib)t} + e_k e^{(a-ib)t}$

$$\begin{aligned}
&(a_k + ib_k)e^{at}(\cos(bt) + i\sin(bt)) + (a_k - ib_k)e^{at}(\cos(bt) - i\sin(bt)) \\
&= e^{at}\left[\cos(bt)(a_k + ib_k + a_k - ib_k) + i\sin(bt)(a_k + ib_k - a_k + ib_k)\right] \\
&= e^{at}\left[\cos(bt)2a_k + i\sin(bt)2ib_k\right] \\
&= 2e^{at}\left[a_k\cos(bt) - b_k\sin(bt)\right] \\
&= 2a_k e^{at}\cos(bt) - 2b_k e^{at}\sin(bt)
\end{aligned}$$

where a_k and b_k are constants. ■

This occurrence of complex conjugate coefficients (as well as of the roots of the denominators) will be illustrated in example 2 later in this chapter.

Lemma 7. *In the absence of any Erlang densities, the unconditional sojourn time density function of a queue with N modulation states is a mixture of N_1 exponential (e^{at}) densities, one of which is a constant of the form $c_0 e^{0t} = c_0$ and N_2 densities of types $e^{at}\cos(bt)$ and $e^{at}\sin(bt)$, where $N_1 + 2N_2 = N^2 + N + 1$.*

Proof

When there are no Erlang densities, the roots x_i of $v(s)$ are distinct and $m_k = 1$ for all i, k in expression (4.7). Therefore, $F(t)$ is a mixture of exponential densities as well as densities of the form $e^{at}\cos(bt)$ and $e^{at}\sin(bt)$. The generating function $\overrightarrow{\mathbf{D}}(z, s)$ is evaluated for particular values of z, which are the eigenvalues ξ_i found during spectral expansion. In a N-state modulated infinite queue there are N distinct eigenvalues (within the unit circle). Now, N of the roots of the denominator of $\overrightarrow{\mathbf{D}}(z, s)$ depend on z, while the other $N + 1$ do not. The eigenvalues ξ_i are distinct in general, so that taken together $\overrightarrow{\mathbf{D}}(\xi_1, s), \ldots, \overrightarrow{\mathbf{D}}(\xi_N, s)$ contribute $N^2 + N + 1$ distinct denominator-roots and hence distributions. The $e^{at}\cos(bt)$ and $e^{at}\sin(bt)$ densities are caused by a pair of complex conjugate roots, so that they also only occur in pairs.

The addition of $\overrightarrow{\mathbf{L}}_0(s)$ does not give any additional exponentials, because its denominator shares its roots with those found in $\overrightarrow{\mathbf{b}}(z, s)$, which are already included by $\overrightarrow{\mathbf{D}}(z, s)$. ■

4.4 Example 1: Exponential Distributions

Our first example demonstrates the solution process on a $c = 3$ server infinite queue with $N = 2$ modulation states and modulation matrix

$$\underline{\mathbf{Q}} = \begin{pmatrix} -2 & 2 \\ 1 & -1 \end{pmatrix}$$

The arrival process is given by $\underline{\mathbf{\Lambda}} = \underline{\mathbf{diag}}(2, 4)$, $\underline{\mathbf{\Theta}} = \underline{\mathbf{diag}}(1/10, 1/10)$, negative customers arrive according to $\underline{\mathbf{K}} = \underline{\mathbf{diag}}(1/2, 1/2)$, $\underline{\mathbf{R}} = \underline{\mathbf{diag}}(2/10, 2/10)$ and services occur with parameters $\underline{\mathbf{M}} = \underline{\mathbf{diag}}(1, 1)$ and $\underline{\mathbf{\Phi}} = \underline{\mathbf{diag}}(1/10, 1/10)$.

4.4.1 Calculating L0

Using the formulae from section 4.2, we find

$$\overrightarrow{\mathbf{L}}_0(s) = \underline{\mathbf{A}}_{L0}^{-1}(s)\underline{\mathbf{M}}\overrightarrow{\mathbf{e}}/s$$

Where

$$\underline{\mathbf{A}}_{L0}(s) = s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \underline{\mathbf{K}}/c$$

$$\underline{\mathbf{A}}_{L0}(s) = \begin{pmatrix} s+2+1+\frac{1}{6} & -2 \\ -1 & s+1+1+\frac{1}{6} \end{pmatrix} = \begin{pmatrix} s+\frac{19}{6} & -2 \\ -1 & s+\frac{13}{6} \end{pmatrix}$$

To calculate $\underline{\mathbf{A}}_{L0}^{-1}(s)$, we need its determinant and cofactors. As it is a 2×2 matrix, the cofactors are simple to determine. We also find $\det \underline{\mathbf{A}}_{L0} = (s+\frac{7}{6})(s+\frac{25}{6})$, so that

$$\underline{\mathbf{A}}_{L0}^{-1}(s) = \frac{1}{(s+\frac{7}{6})(s+\frac{25}{6})} \begin{pmatrix} s+\frac{15}{6} & 2 \\ 1 & s+\frac{19}{6} \end{pmatrix}$$

Hence, we find

$$\begin{aligned} \overrightarrow{\mathbf{L}}_0(s) &= \frac{1}{(s+\frac{7}{6})(s+\frac{25}{6})} \begin{pmatrix} s+\frac{13}{6} & 2 \\ 1 & s+\frac{19}{6} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} /s \\ &= \frac{1}{s(s+\frac{7}{6})(s+\frac{25}{6})} \begin{pmatrix} s+\frac{25}{6} \\ s+\frac{25}{6} \end{pmatrix} \end{aligned}$$

Our particular values of $\underline{\mathbf{K}}$ and $\underline{\mathbf{M}}$ allow for the $(s+\frac{25}{6})$-factor to cancel[2], so that

$$\overrightarrow{\mathbf{L}}_0(s) = \frac{1}{s(s+\frac{7}{6})}\overrightarrow{\mathbf{e}}$$

Using partial fractions we find

$$\overrightarrow{\mathbf{L}}_0(s) = \left(\frac{\frac{6}{7}}{s} - \frac{\frac{6}{7}}{s+\frac{7}{6}}\right)\overrightarrow{\mathbf{e}}$$

It can therefore be seen that the Laplace transform for the processing region will contribute an exponential distribution $c_1 e^{-\frac{7}{6}t}$ as well as a constant term $c_0 e^{0t} = c_0$ to the solution for $F(t)$. For a more complex choice of queueing parameters, cancellation of one of the roots of the determinant, as has happened here, would not be possible.

4.4.2 Calculating b

The evaluation of $\overrightarrow{\mathbf{b}}(z,s)$ requires $\overrightarrow{\mathbf{L}}_0$ and a series of straightforward matrix multiplications by diagonal, s-independent matrices. The roots in s in the denominators of $\overrightarrow{\mathbf{b}}(z,s).\overrightarrow{\mathbf{e}}_i$ are the same as those of $\overrightarrow{\mathbf{L}}_0.\overrightarrow{\mathbf{e}}_i$. Indeed we find,

$$\overrightarrow{\mathbf{b}}(z,s) = \frac{z^3(345 - 6s(z-5) - 65z)}{2s(s+\frac{7}{6})(z-10)(z-5)}\overrightarrow{\mathbf{e}}$$

4.4.3 Calculating C

The repeating region (levels $j \geq c$) is represented using the generating function

$$\overrightarrow{\mathbf{D}}(z,s) = \underline{\mathbf{C}}^{-1}(z,s)\,\overrightarrow{\mathbf{b}}(z,s)$$

where $\underline{\mathbf{C}}(z,s)$ and $\overrightarrow{\mathbf{b}}(z,s)$ are given by (4.3) and (4.4).

By substituting the appropriate values from the queue model we find

$$\underline{\mathbf{C}}(z,s) = \begin{pmatrix} s + \frac{69}{2} + \frac{270}{z-10} + \frac{10}{z-5} & -2 \\ -1 & s + \frac{67}{2} + \frac{270}{z-10} + \frac{10}{z-5} \end{pmatrix}$$

To invert this matrix, we calculate the determinant and adjoint matrix.

$$\det(\underline{\mathbf{C}}(z,s)) = \left(s - \frac{-71z^2 + 505z - 650}{2(z-5)(z-10)}\right)\left(s + \frac{65(z-1)(z-\frac{70}{13})}{2(z-5)(z-10)}\right)$$

[2]This behaviour occurs whenever $\underline{\mathbf{K}}$ and $\underline{\mathbf{M}}$ are multiples of the identity matrix $\underline{\mathbf{I}}$

$$adj(\underline{\mathbf{C}}(z,s)) = \begin{pmatrix} s+\frac{67}{2}+\frac{270}{z-10}+\frac{10}{z-5} & 2 \\ 1 & s+\frac{69}{2}+\frac{270}{z-10}+\frac{10}{z-5} \end{pmatrix}$$

As expected, both roots of the determinant depend on z.

$$\overrightarrow{\mathbf{D}}(z,s) = \frac{z^3(345-6s(z-5)-65z)}{2s(s+\frac{7}{6})(s+\frac{65(z-1)(z-\frac{70}{13})}{2(z-5)(z-10)})(z-5)(z-10)}\overrightarrow{\mathbf{e}}$$

The denominator of the elements of $\overrightarrow{\mathbf{D}}(z,s)$ is a polynomial in s, that shares its roots with those found for $\underline{\mathbf{C}}^{-1}(z,s)$ and $\overrightarrow{\mathbf{B}}(z,s)$. The total number of roots is one less than the sum of the constituent parts because, once again, we were able to cancel one[3] of them due to the simple nature of $\underline{\mathbf{M}}$ and $\underline{\mathbf{K}}$. Had no cancellation been possible throughout the calculations, as is to be expected in the general case, we would be left with polynomials in s of degree $2N+1$ in the denominators of $\overrightarrow{\mathbf{D}}(z,s)$.

In our example, the eigenvalues had to be found numerically during spectral expansion and they are (displaying 6 significant digits only) $\xi_1 = 0.371358$ and $\xi_2 = 0.951724$. The eigenvectors are $\overrightarrow{\psi}_1 = (0.948186, -0.317733)$ and $\overrightarrow{\psi}_2 = (0.438851, 0.900968)$, with eigenmode-coefficients $a_1 = 0.0192014$ and $a_2 = 0.0377927$.

Substituting these values into (4.5) and using partial fractions, the Laplace transform of the unconditional sojourn time distribution is found as

$$L(s) = \frac{0.843596}{s} - \frac{0.900988}{0.189873+s} + \frac{0.0571398}{\frac{7}{6}+s} + \frac{0.000251995}{2.29820+s}$$

After inversion, we find that

$$F(t) = P(T<t) = 0.843596 - 0.900988e^{-0.189873t} + 0.0571398e^{-\frac{7}{6}t} + 0.000251995e^{-2.29820t}$$

From this expression or from figure 4.1, which plots $F(t)$, we see that

$$\lim_{t\to\infty} P(T<t) = 0.843596$$

This is the probability that a customer is not killed. To obtain the probability density function, $f(t)$, of the sojourn time of a customer *given that it is not killed*, we divide $F(t)$ by this probability, and differentiate with respect to t. This probability density is shown in figure 4.2. It is not monotonic, so that the most likely sojourn time is not zero, but approximatively 1.245.

[3] $\det(\underline{\mathbf{C}}(z,s))$ had a factor $\left(s - \frac{-71z^2+505z-650}{2(z-5)(z-10)}\right)$

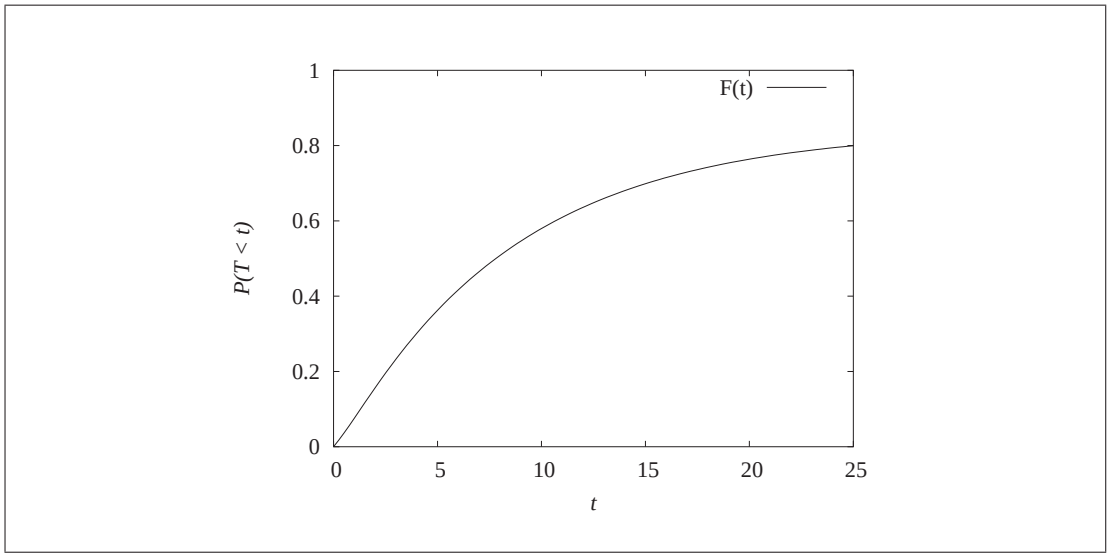


Figure 4.1: The Sojourn time distribution for our first example queue. The probability does not reach 1, because a killed customer never completes service.

4.5 Example 2: sin and cos-exponential Distributions

For this example we consider a queue whose solution involves complex conjugate roots, which lead to the presence of densities of the form $e^{at}\cos(bt)$ - in the sojourn time distribution. To keep calculations simple we will not go into as much detail as before and use a queue with no batches or negative customers[4]. The parameters are $c = 1$, $L = \infty$, $N = 3$ and the modulation process is given by

$$\underline{\mathbf{Q}} = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$

the arrival process is given by $\underline{\mathbf{\Lambda}} = \underline{\mathbf{I}}$ and services occur with rates 2, $3/2$ and 0 respectively in phases 1, 2 and 3, *i.e.*

$$\underline{\mathbf{M}} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

[4]We are re-using the queue from section 3.4.1 for $k = 1$

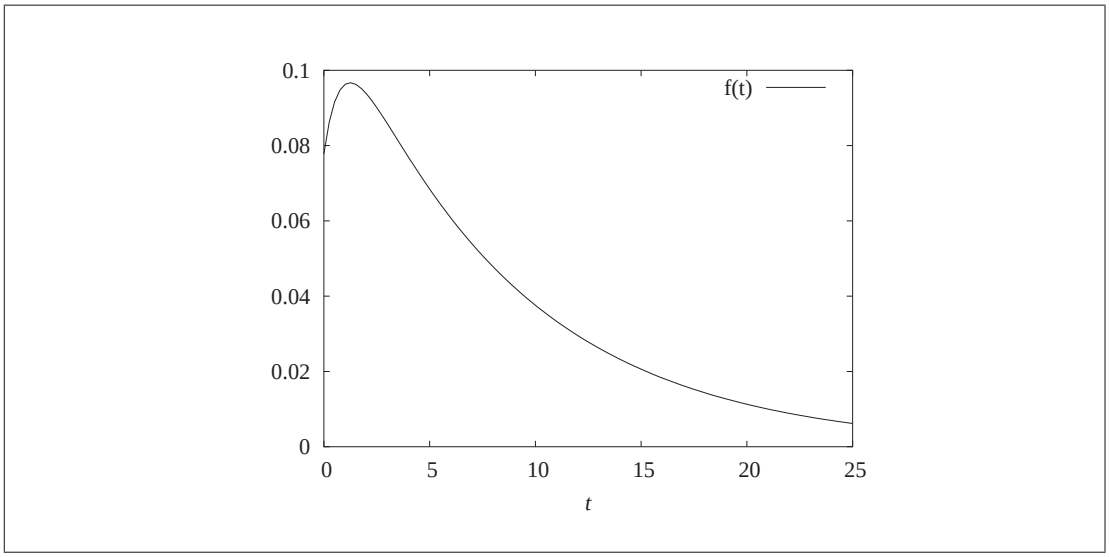


Figure 4.2: Probability density of the customer sojourn time for example one, if it is not killed.

It is found that

$$\overrightarrow{\mathbf{L}}_0 = \frac{1}{p(s)} \begin{pmatrix} (1+s)(13+4s) \\ 13+3s(4+s) \\ 13+4s \end{pmatrix}$$

where $p(s) = s(2s^3 + 13s^2 + 26s + 13)$, or approximatively

$$p(s) \approx 2s(s+0.746844)(s+2.876578+0.654667i)(s+2.876578-0.654667i)$$

Further,

$$\overrightarrow{\mathbf{D}}(z,s) = -\overrightarrow{\mathbf{L}}_0 + \frac{1}{q(s)} \begin{pmatrix} 13+17s+4s^2-6z-6sz \\ 13+12s+3s^2-6z-6sz \\ 13+4s-6z \end{pmatrix}$$

where

$$q(s) = s(13+26s+13s^2+2s^3-19z-26sz-7s^2+6z^2+6sz^2)$$

To 6 digits precision, the three eigenvalues were $\xi_1 = 0.271723 + 0.0653162i$, $\xi_2 = 0.271723 - 0.0653162i$ and $\xi_3 = 0.887580$. The corresponding eigenvectors are

$$\overrightarrow{\mathbf{\Psi}}_1 = \begin{pmatrix} 0.583421 \\ -0.324794 + 0.626038i \\ -0.0132018 - 0.411712i \end{pmatrix} \quad \overrightarrow{\mathbf{\Psi}}_2 = \begin{pmatrix} 0.583421 \\ -0.324794 - 0.626038i \\ -0.0132018 + 0.411712i \end{pmatrix}$$
$$\overrightarrow{\mathbf{\Psi}}_3 = \begin{pmatrix} 0.565626 \\ 0.542985 \\ 0.620672 \end{pmatrix}$$

And the coefficients are $a_1 = 0.00850742 - 0.0170986i$, $a_2 = 0.00850742 + 0.0170986i$ and $a_3 = 0.0633765$. With these quantities, we use formula (4.6) to find the desired $L(s)$, which, after applying partial fractions, is given by

$$L(s) = \frac{1}{s} - \frac{0.985204}{s + 0.125166} - \frac{0.00739775 + 0.00398902i}{s + 2.47918 - 0.836319i} - \frac{0.00739775 - 0.00398902i}{s + 2.47918 + 0.836319i}$$

As expected, the coefficients of the complex roots in the partial fractions expansion are complex conjugate. The inversion to generate the unconditional sojourn time is now straightforward, *i.e.*

$$F(t) = 1 - 0.985204e^{-0.125166} - 2 \times 0.00739775e^{-2.47918t}\cos(0.836319t) + 2 \times 0.00398902e^{-2.47918t}\sin(0.836319t)$$
$$f(t) = \frac{\partial F(t)}{\partial t} = 0.1233137e^{-0.125166t} + 0.0433529e^{-2.47918t}\cos(0.836319t) - 0.00740526e^{-2.47918t}\sin(0.836319t)$$

As there was no killing in this queue, $F(t) = P(T < t)$ reaches 1 as $t \to \infty$, and we have a cumulative distribution function. The presence of oscillating $\cos$ and $\sin$ terms in such a distribution function is unusual, because it is conceivable that $F(t)$ could be non-monotonic. The coefficients that are encountered are small however, and hence $F(t)$ is dominated by the exponential distribution, ensuring monotonicity. Figure 4.3 shows the contribution of the $\cos$ and $\sin$ terms to $f(t)$. The oscillations are heavily damped by the $e^{-2.47918t}$ term present in both functions.

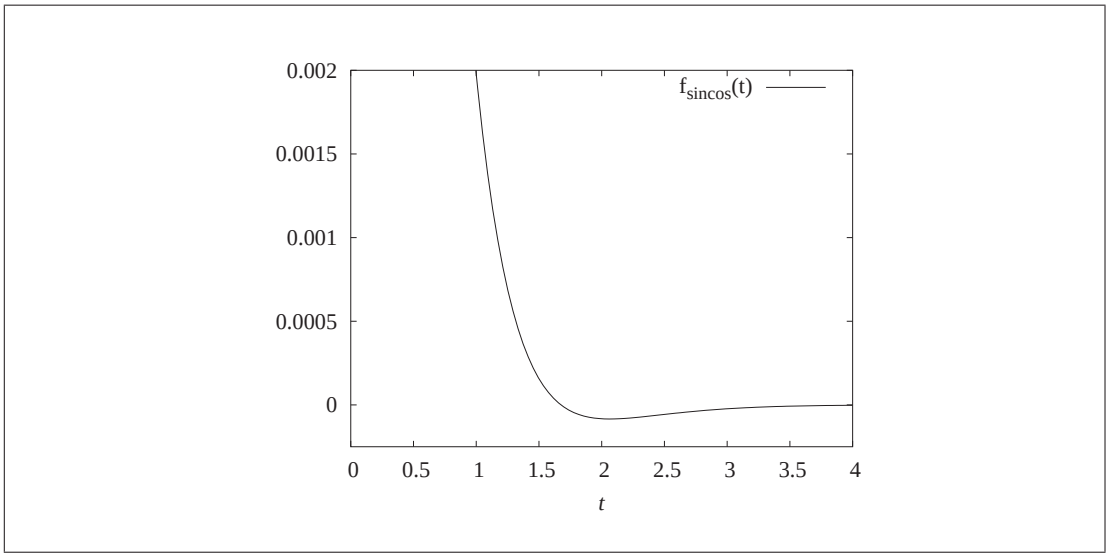


Figure 4.3: Oscillating components of the sojourn time probability density in the second example. Note that $f_{sincos}(0) \approx 0.0433$

4.6 Example 3: Erlang-2 Distributions

In general it is quite unlikely to find double or even higher order roots in the denominators of the elements of $\overrightarrow{\mathbf{D}}(z,s)$. The inversion step would then produce Erlang-n distributions within the unconditional sojourn time distribution $F(t)$. Through careful choice of parameters, we can contrive a case where this happens. Consider the expression for $\overrightarrow{\mathbf{D}}(z,s)$ found for the first example.

$$\overrightarrow{\mathbf{D}}(z,s) = \frac{z^3(345 - 6s(z-5) - 65z)}{2s(s+\frac{7}{6})(s+\frac{65(z-1)(z-\frac{70}{13})}{2(z-5)(z-10)})(z-5)(z-10)}\overrightarrow{\mathbf{e}}$$

For an appropriate choice of z, we can force the z-dependent root in s of the denominator to coincide with the z-independent root $s = -\frac{7}{6}$. Solving the equation $\frac{65(z-1)(z-\frac{70}{13})}{2(z-5)(z-10)} = \frac{7}{6}$ gives the two solutions

$$z_1 = \frac{5}{94}(57 - \sqrt{1933}) \approx 0.693303$$

and

$$z_2 = \frac{5}{94}(57 + \sqrt{1933}) \approx 5.370526$$

If one of the eigenvalues we find during the queue solution process is equal to either of these, we have a double root at $s = \frac{7}{6}$. As we are dealing with infinite queues

only, the eigenvalue 5.370526 is of course not attainable. The challenge now lies in finding appropriate parameters for our queue that lead to an eigenvalue $\xi = 0.693303$, without changing $\overrightarrow{\mathbf{D}}(z,s)$. This problem is not directly or symbolically soluble, but it can be approached by varying the arrival rate at the queue through multiplying the rate matrix $\underline{\mathbf{\Lambda}}$ by a factor f. An increase in $f\underline{\mathbf{\Lambda}}$ causes an increase in the queue load and (in general) eigenvalues. At the same time the expression $\overrightarrow{\mathbf{D}}(z,s)$ is independent of $\underline{\mathbf{\Lambda}}$, so we can use a simple bracketing algorithm to find an arbitrarily close approximation to a particular value of $f\underline{\mathbf{\Lambda}}$ that gives the desired eigenvalue. It has been found that a value of $f \approx 0.6587182746986538$ yields a queue solution with eigenvalues $\xi_1 = 0.291041$ and $\xi_2 = 0.693303$. As stated before, the value of $\overrightarrow{\mathbf{D}}(z,s)$ is independent of this change in the arrival rate and therefore identical to that given in example 1. Using the expressions for $L(s)$ and partial fractions, we get

$$L(s) = \frac{0.85044498}{s} + \frac{0.000633749}{s + 2.567019} - \frac{0.851078}{s + \frac{7}{6}} - \frac{0.420691}{(s + \frac{7}{6})^2}$$

Using the inversion formulae the sojourn time distribution is

$$F(t) = 0.85044498 + 0.000633749e^{-2.567019t} - 0.851078e^{-\frac{7}{6}t} - 0.420691te^{-\frac{7}{6}t}$$

with the associated pdf being

$$f(t) = \frac{\partial \hat{F}(t)}{\partial t} = -0.00191293e^{-2.567019t} + 0.672865e^{-\frac{7}{6}t} + 0.5771171te^{-\frac{7}{6}t}$$

where $\hat{F}(t) = \frac{F(t)}{0.85044498}$ is the re-normalised cumulative distribution function that is not conditioned on a customer being killed. One of the components of the probability density function $f(t)$ is an Erlang-2 distribution — this component is illustrated in figure 4.4.

4.7 Introducing multiple streams

To eventually use the response time distribution in a network setting, we need to extend the formulae to allow for multiple streams. Most important is the inclusion of multiple positive and negative customer streams, which occurs when a number of queue outputs

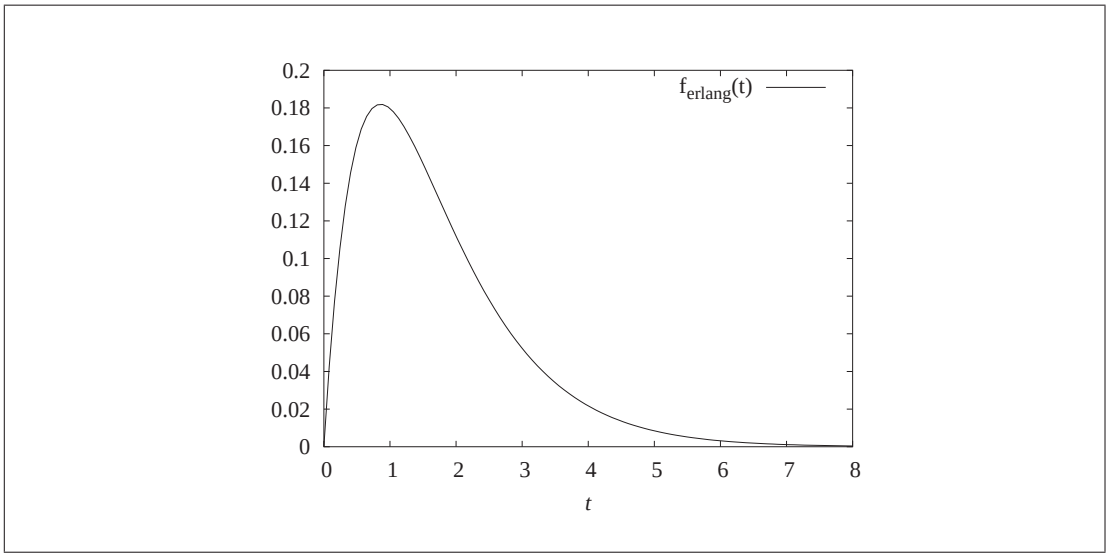


Figure 4.4: Erlang component of the sojourn time probability density for example 3.

are fed into a queue. Multiple service streams, whilst it is not immediately obvious what role they would have in a network, are also going to be included in our derivation, due to the simplicity of the changes necessary to accommodate them.

4.7.1 Multiple positive arrival streams

The effect of positive arrivals on the sojourn time is very limited, because, using the RCH killing paradigm, we only need to keep track of customers *in front* of the tagged customer. Positive arrivals join the queue *behind* the tagged customer, so the inclusion of multiple positive arrival streams is straightforward.

The changes to expression (4.5) only involve summing over all arrival streams, with appropriate adjustment of the total arrival flux, λ^*. When the positive arrival rate matrices for each stream are given by $\underline{\mathbf{\Lambda}}_i$, and batch matrices by $\underline{\mathbf{\Theta}}_i$, we have

$$\lambda^* = \sum_{h=1}^{N} \pi_h \left(\sum_{k=1}^{n^{arr}} (\underline{\mathbf{\Lambda}}_k)_{h,h} \right)$$

$$L(s) = \frac{1}{\lambda^*} \sum_{i=1}^{n^{arr}} \left[(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}_i) \overrightarrow{\mathbf{L}}_0(s) \sum_{j=0}^{c-1} \left(\sum_{q=0}^{j} \underline{\mathbf{\Theta}}_i^{j-q} \underline{\mathbf{\Lambda}}_i \overrightarrow{\mathbf{v}}_q \right) \right. \tag{4.8}$$
$$\left. -(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}_i)\underline{\mathbf{\Theta}}_i^{-1} \sum_{k=1}^{n^*} a_k \xi_k (\underline{\mathbf{I}} - \xi_k \underline{\mathbf{\Theta}}_i^{-1})^{-1} \underline{\mathbf{\Lambda}}_i \overrightarrow{\psi}_k \overrightarrow{\mathbf{D}}(\xi_k, s) \right]$$

Note that n^*, the number of eigenvalues present, can now be larger than N.

4.7.2 Multiple negative arrival streams

The inclusion of multiple killing streams is done by augmenting the killing rate matrix $\underline{\mathbf{K}}$ and its associated batch size distribution matrix $\underline{\mathbf{R}}$ using a sum over the n^{kill} negative arrival streams.

Within the processing region, the Laplace densities become

$$\overrightarrow{\mathbf{L}}_0(s) \equiv \overrightarrow{\mathbf{L}}_{c-1}(s) = \left(s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \underline{\mathbf{M}} + \frac{1}{c} \sum_{k=0}^{n^{kill}} \underline{\mathbf{K}}_k \right)^{-1} \underline{\mathbf{M}} \overrightarrow{\mathbf{e}}/s$$

As before, we use a generating function $\overrightarrow{\mathbf{D}}(z,s) = \sum_{j=c}^{\infty} \overrightarrow{\mathbf{L}}_j z^j$ for representing the repeating region. Its value is determined by the equation

$$\overrightarrow{\mathbf{D}}(z,s) = \underline{\mathbf{C}}^{-1}(z,s) \overrightarrow{\mathbf{B}}(z,s) \tag{4.9}$$

where

$$\begin{aligned}
\underline{\mathbf{C}}(z,s) &= s\underline{\mathbf{I}} - \underline{\mathbf{Q}} + \textstyle\sum_{k=0}^{n^{kill}} \underline{\mathbf{K}}_k + c\underline{\mathbf{M}} - \sum_{k=1}^{n^{kill}} \underline{\mathbf{K}}_k (\underline{\mathbf{I}} - \underline{\mathbf{R}}_k)(\underline{\mathbf{I}} - \underline{\mathbf{R}}_k z)^{-1} z \\
&- c\underline{\mathbf{M}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}} z)^{-1} z \\
\overrightarrow{\mathbf{b}}(z,s) &= (cz^c/s)\underline{\mathbf{M}}\underline{\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}} z)^{-1} \overrightarrow{\mathbf{e}} \\
&+ \left(\textstyle\sum_{k=0}^{n^{kill}} \underline{\mathbf{K}}_k (\underline{\mathbf{I}} - \underline{\mathbf{R}}_k)(\underline{\mathbf{I}} - \underline{\mathbf{R}}_k z)^{-1} + c\underline{\mathbf{M}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}} z)^{-1} \right) \\
&\times (cz^c) \overrightarrow{\mathbf{L}}_0(s)
\end{aligned} \tag{4.10}$$

4.7.3 Multiple service streams

Once multiple negative customer streams are accommodated, the addition of multiple service streams is straightforward. When the killing paradigm is RCH, the arrival of

a negative customer batch is indistinguishable, to all but the tagged customer, from a batch service event with the same parameters. It is therefore possible to re-label all but one of the n^{serv} service streams to negative customer streams. The resulting system has one service stream (with rate matrix $\underline{\mathbf{M}}$ and batch matrix $\underline{\mathbf{\Phi}}$) as well as $n^{kill} + n^{serv} - 1$ negative customer streams, consisting of the original as well as the relabelled ones.

The expressions for multiple negative customer streams derived in the previous section can now be used to derive the sojourn time distribution conditional on the tagged customer completing service in the one remaining service stream. The unconditional sojourn time distribution is given by simply re-labelling the specified service streams so that each, in turn, takes the place of the single service stream.

4.8 Automation of the sojourn time calculation

When used in conjunction with the automated generation of steady-state probabilities using spectral expansion, the sojourn time calculations lend themselves to a straightforward automated implementation. The use of spectral expansion as opposed to other matrix geometric methods is imperative, as the eigenvalues found during this process are subsequently used for the values of z in the generating function $\overrightarrow{\mathbf{D}}(z, s)$, and would not be immediately available were another matrix geometric method being used. The algorithm to calculate the sojourn time of a geometrically batched infinite queue is then specified as follows

- Use the spectral expansion method to derive the steady-state probability distribution of the queue. The probabilities of the infinite repeating region are given in terms of eigenvalues and eigenvectors.
- Generate the vector generating function $\overrightarrow{\mathbf{D}}(z, s)$ using (4.9) and (4.10).
- Using the found quantities, calculate the unconditional Laplace transform $L(s)$ using the expression (4.5).
- Perform symbolic partial fractions to transform $L(s)$ to a sum of simple fractions.
- Pattern match each fraction to get the distribution and hence density function.

Clearly, the computational cost is dominated by the derivation of the steady-state probability distribution, which requires the finding of eigenvalues of a matrix. Finding $L(s)$ involves multiplication by diagonal and inversion of positive-definite $N \times N$ matrices, which is both efficient and numerically stable. Finally, the partial fraction step is done by solving a small linear system for the coefficients c_i.

Due to the availability of our *BlueSolver* package written in Mathematica®that automates the spectral expansion portion of the work, we have implemented the remainder of the steps using the same environment, which has proven itself to be efficient and stable.

Chapter 5

Networks

5.1 Introduction

In this chapter we describe and solve systems containing networks of queues of the type described in chapter 2 using methods from chapter 3. Within these networks, the output of any one queue is redirected to the input of another queue as either positive or negative customers (or both, probabilistically split). As described in chapter 2, batch arrivals to a finite queue are necessarily truncated to its maximal queue length L. When an arrival batch is truncated in this manner, customers are lost. Both our analytical approximation techniques and the simulators that follow do not allow a sending queue to be blocked by a receiving queue.

Each queue is assigned identifier Q_i, which are then putting into a list $\{Q_1, Q_2, Q_3\}$, which allows us to additionally identify each queue by its position in the list. The routing for this enumerated list is then described by way of a square probabilistic matrix $\underline{\mathbf{Y}}$, which is invariant over time. The probability distribution of the destinations for a departing batch of customers from queue Q_i is given by the i^{th} column, $\overrightarrow{\mathbf{y}}_i$. Entries of $\underline{\mathbf{Y}}$ are non-negative and the sum of each column is 1, *i.e.* $\left\|\overrightarrow{\mathbf{y}}_i\right\|_1 = 1$. Figure 5.1 shows a simple network containing three queues with a routing matrix given

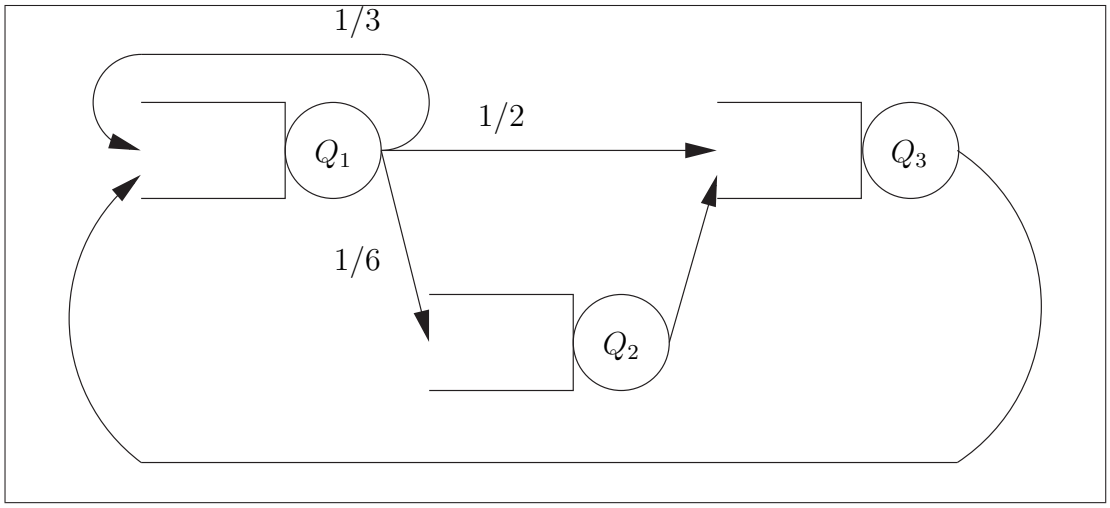


Figure 5.1: A closed network with routing probabilities.

by

$$\underline{\mathbf{Y}} = \begin{pmatrix} \frac{1}{3} & 0 & 1 \\ \frac{1}{6} & 0 & 0 \\ \frac{1}{2} & 1 & 0 \end{pmatrix}$$

Entries on the diagonal correspond to departing customers that are fed back into the same queue. For example, served customers from queue Q_1 in the example closed network shown in figure 5.1 re-join with probability $\frac{1}{3}$.

The network is called closed, because customers inside the system cannot leave and customers cannot enter from the outside. To describe open networks, in which external customers can enter and leave, we provide *Sources* and *Sinks*, which generate and remove customers respectively. For the matrix $\underline{\mathbf{Y}}$, the incorporation of Sources and Sinks is straightforward. Both types behave just as a queue for routing purposes, so that they are added to the enumeration $\{Source, Q_1, Q_2, Q_3, Sink\}$ and each has a column associated with it in the routing matrix $\underline{\mathbf{Y}}$. Due to their particular nature, it does not make sense to send customers to a Source or have them incoming from a Sink. We therefore have the restriction that if object i is a Source, then the i^{th} row of $\underline{\mathbf{Y}}$ is a zero vector. Similarly, if object i is a Sink, then the i^{th} column of $\underline{\mathbf{Y}}$ is a zero vector. When Source and Sink are added to the previous network as in figure 5.2, the

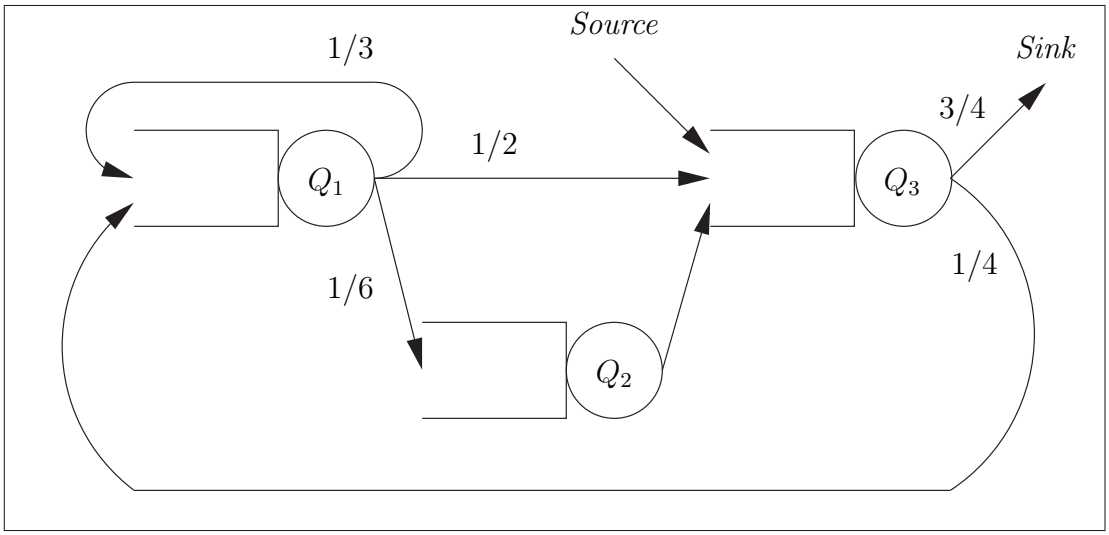


Figure 5.2: An open network: the previous network with a Source and Sink added.

routing matrix becomes

$$\underline{\mathbf{Y}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{6} & 0 & 0 & 0 \\ 1 & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{4} & 0 \end{pmatrix}$$

5.2 PEPA representation

Performance Evaluation Process Algebra, henceforth PEPA [Hil94] is a stochastic process algebra that is well suited to representing the important concepts found for the interaction between queues, $Sources$ and $Sinks$. An overview of its syntax is given in Appendix B. Though it is helpful to describe customer interactions in queues it should not be used to find steady-state probabilities by way of tools such as the PEPA workbench [GilHil94]. This is because the workbench tries to directly solve the entire underlying Markov chain, which leads to a state space explosion and hence we specifically avoid this by using spectral expansion or other matrix geometric methods are used.

At this point, our aim is not to generate closed analytical expressions, the balance

equations derived in chapter 2 do this. Instead, we lay out the mechanics of customers being passed from one queue to another in a rigorous manner.

A $Source$ that generates customers can be written as the following.

$$Source \stackrel{def}{=} (extarr, \mu).Source$$

It carries out the action of type $extarr$ with exponential rate μ and then continues behaving as $Source$. Similarly, the following PEPA description of a Sink allows for the $leave$ action to happen at the passive rate $\top$, *i.e.* at a rate dictated by the process with which $Sink$ is cooperating.

$$Sink \stackrel{def}{=} (leave, \top).Sink$$

Our general queue, Q, with tail-vulnerable (or head-per) negative customers is defined by

$$\begin{aligned} Q_0 &\stackrel{def}{=} (arrival, \top).Q_1 + (kill, \top).Q_0 \\ Q_i &\stackrel{def}{=} (arrival, \top).Q_{i+1} + (kill, \top).Q_{i-1} + (service, \mu).Q_{i-1} \\ Q_L &\stackrel{def}{=} (arrival, \top).Q_L + (kill, \top).Q_{L-1} + (service, \mu).Q_{L-1} \end{aligned}$$

A description for tail-safe negative customers is

$$\begin{aligned} Q_0 &\stackrel{def}{=} (arrival, \top).Q_1 + (kill, \top).Q_0 \\ Q_1 &\stackrel{def}{=} (arrival, \top).Q_2 + (kill, \top).Q_1 + (service, \mu).Q_0 \\ Q_i &\stackrel{def}{=} (arrival, \top).Q_{i+1} + (kill, \top).Q_{i-1} + (service, \mu).Q_{i-1} \\ Q_L &\stackrel{def}{=} (arrival, \top).Q_L + (kill, \top).Q_{L-1} + (service, \mu).Q_{L-1} \end{aligned}$$

Blocking behaviour and losses

In the model for Q (*i.e.* Q_i for any chosen $0 \leq i \leq \infty$) given above, the two passive activities (indicated by $\top$ as rate parameter) $arrival$ and $kill$ are always enabled. When Q is involved in a cooperation (described below), with another agent, P say, $Q \underset{\{arrival,kill\}}{\bowtie} P$, it cannot block any arrivals (positive or negative). Indeed, arrivals to a full queue are lost in the sense that there is no change of state. This means that a receiving queue cannot affect the service activity of a sending queue. Similarly, a

negative arrival to an empty queue is also dropped and has no influence on the queue length.

Sometimes blocking behaviour is desired, in which case we use the following description for the behaviour of Q.

$$\begin{aligned} Q_0 &\stackrel{def}{=} (arrival, \top).Q_1 \\ Q_1 &\stackrel{def}{=} (arrival, \top).Q_2 + (service, \mu).Q_0 \\ Q_i &\stackrel{def}{=} (arrival, \top).Q_{i+1} + (kill, \top).Q_{i-1} + (service, \mu).Q_{i-1} \\ Q_L &\stackrel{def}{=} (kill, \top).Q_{L-1} + (service, \mu).Q_{L-1} \end{aligned}$$

Here, arrivals are not enabled when the queue is full (length L) and negative customers are blocked at $j = 0$ and $j = 1$, so that this models tail-safe (t^s) behaviour.

Modulation

The modulation process of a queue (*c.f.* chapter 2.2) is an independent Markov chain that selects the service rates that the queue experiences. Similarly, the modulation process of a $Source$ chooses the rates at which customers join the network. To allow the modulation process to affect the rates of the queue, we define a modulation process Mod and put it into a cooperation with the queue. All the necessary transition rates can be taken directly from the modulation matrix $\underline{\mathbf{Q}}$. It follows an example for a $N = 3$-state modulated system.

$$\begin{aligned} Mod_1 &\stackrel{def}{=} (modulate, q_{12}).Mod_2 + (modulate, q_{13}).Mod_3 + (service, \mu_1).Mod_1 \\ Mod_2 &\stackrel{def}{=} (modulate, q_{21}).Mod_1 + (modulate, q_{23}).Mod_3 + (service, \mu_2).Mod_2 \\ Mod_3 &\stackrel{def}{=} (modulate, q_{31}).Mod_1 + (modulate, q_{32}).Mod_2 + (service, \mu_3).Mod_3 \end{aligned}$$

And more generally,

$$Mod_i \stackrel{def}{=} \sum_{j \neq i}^{N} (modulate, q_{ij}).Mod_j + (service, \mu_i).Mod_i$$

A modulated G-Queue with arrivals that come from synchronisation with another process (*e.g.* a $Source$ of external arrivals, not shown in this expression) is now given by

$$Q_0 \underset{\{service\}}{\bowtie} Mod_1$$

where the Q_i-process has its service rates all set to $\top$.

Sources can also be modulated using the same mechanism. The $Source$ definition in (5.1) will "serve" customers at rates μ_1, μ_2 or μ_3, depending on the modulation state.

$$Source \stackrel{def}{=} (service, \top).Source \tag{5.1}$$

$$Source \underset{\{service\}}{\bowtie} Mod_1$$

Customer Passing

The linking of the output of one queue to the input of another is achieved using a cooperation of the two. We relabel the activity name of the passive action to match that of the active action and then include the joint activity in the cooperation set. Figure 5.3

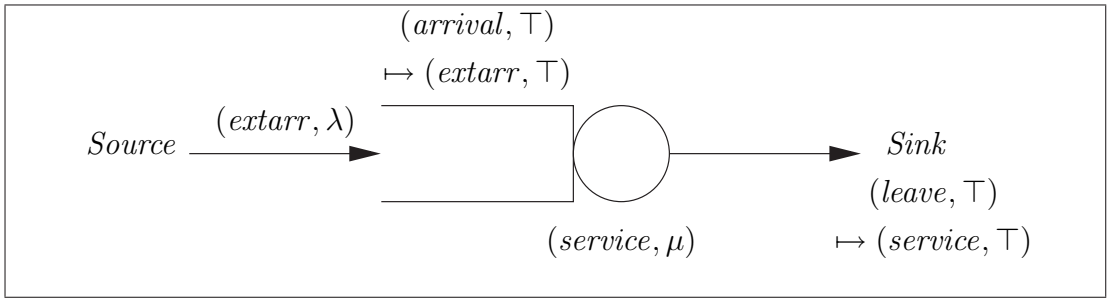


Figure 5.3: Relabelling of actions in a cooperation

shows a situation where two activities are relabelled. The source feeds customers into the queue, requiring the $extarr$ action of the source and the $arrival$ action of the queue be synchronised. We therefore map $(arrival, \top) \mapsto (extarr, \top)$ within the Queue Q_0 and its derivatives. Similarly, the $service$ activity of the queue coincides with the $leave$ activity of the $Sink$ and hence we have $(leave, \top) \mapsto (service, \top)$ for the $Sink$. After relabelling, the activity names in a cooperation are not necessarily as intuitive as those in the stand-alone models. This is especially the case for large models and should confusion arise, it is always possible to assign an additional, clarifying label to synchronous actions to clarify its function.

The full system description of the situation in figure 5.3 is now given by

$$Source \underset{\{extarr\}}{\bowtie} \left(Q_0 \underset{\{service\}}{\bowtie} Mod_1\right) \underset{\{service\}}{\bowtie} Sink$$

Geometric Batches

With geometrically distributed batches, queue length changes can potentially traverse the entire range of queue lengths. For each service batch size s we allocate a unique parameterised activity $service_s$ and use the appropriate transition rates. $arrival_s$ and $kill_s$ serve the same purpose for arrival and killing transitions with the exception that their rates are dictated by outside parameters and hence the transitions are passive. Truncation of the passive actions is simulated by the addition of transitions to the truncation levels Q_0 and Q_L via all possible batch sizes above a certain value for positive and negative arrivals.

The full PEPA description for a non-blocking, unmodulated queue with geometrically batched service completions and tail-vulnerable negative customers is

$$
\begin{aligned}
Q_0 &\stackrel{def}{=} \sum_{s=1}^{L}(arrival_s, \top).Q_s + \sum_{s=L+1}^{\infty}(arrival_s, \top).Q_L + \sum_{s=1}^{\infty}(kill_s, \top).Q_0 \\
Q_i &\stackrel{def}{=} \sum_{s=1}^{L-i}(arrival_s, \top).Q_{i+s} + \sum_{s=L-i+1}^{\infty}(arrival_s, \top).Q_L \\
&\quad + \sum_{s=1}^{i}(kill_s, \top).Q_{i-s} + \sum_{s=i+1}^{\infty}(kill_s, \top).Q_0 \\
&\quad + \sum_{s=1}^{i-1}(service_s, (1-\phi)\phi^{s-1}\mu).Q_{i-s} + (service_i, \phi^i).Q_0
\end{aligned}
$$

Correspondingly, we also have geometrically batched sources

$$Source \stackrel{def}{=} \sum_{s=1}^{\infty}(extarr_s, (1-\phi)\phi^{s-1}\mu).Source$$

Combining all features

The inclusion of both geometric batches and Markov modulation is somewhat intricate, because the modulation process that specifies the rate with which service events occur, does not know at which point a batch is truncated. None of the modulation processes actions are allowed to be blocked and hence we need to include a range of "dummy" terms in the transition structure of Q. Sources never truncate and are

unaffected by this.

$$\begin{aligned} Mod_i^{ACTIVITY} &\stackrel{def}{=} \textstyle\sum_{j\neq i}^{N}(modulate, q_{ij}).Mod_j^{ACTIVITY} \\ &\quad + \textstyle\sum_{s=1}^{\infty}(ACTIVITY_s, \mu_s(1-\phi)\phi^{s-1}).Mod_i^{ACTIVITY} \\ Source &\stackrel{def}{=} \textstyle\sum_{s=1}^{\infty}(extarr_s, \top).Source \end{aligned}$$

Now $Source \underset{\{extarr_s\}}{\bowtie} Mod_1^{extarr}$ behaves like a Markov modulated, geometrically batched source.

$$\begin{aligned} Q_0 &\stackrel{def}{=} \sum_{s=1}^{L}(arrival_s, \top).Q_s + \sum_{s=L+1}^{\infty}(arrival_s, \top).Q_L + \sum_{s=1}^{\infty}(kill_s, \top).Q_0 \\ Q_i &\stackrel{def}{=} \sum_{s=1}^{L-i}(arrival_s, \top).Q_{i+s} + \sum_{s=L-i+1}^{\infty}(arrival_s, \top).Q_L \\ &\quad + \sum_{s=1}^{i}(kill_s, \top).Q_{i-s} + \sum_{s=i+1}^{\infty}(kill_s, \top).Q_0 \\ &\quad + \sum_{s=1}^{i-1}(service_s, \top).Q_{i-s} + \sum_{s=i}^{\infty}(service_j, \top).Q_0 \end{aligned}$$

With these definitions it is now possible to give the model of a Markov modulated queue with Markov modulated positive and negative customers and geometric batched transitions. This is subtly different from how such an individual queue in chapter 2 operates in that the modulation processes are *independent* of each other, rather than having a single process selecting the rates for arrivals and services. Such independent modulation is what is encountered in networks, so that this is a desirable feature.

$$\begin{gathered} \left[\left(PSource \underset{\{arrival\}}{\bowtie} Mod_1^{arrival}\right) \underset{\emptyset}{\bowtie} \left(NSource \underset{\{kill\}}{\bowtie} Mod_1^{kill}\right)\right] \\ \underset{\{arrival,kill\}}{\bowtie} \\ \left[\left(Q_0 \underset{\{service\}}{\bowtie} Mod_1^{service}\right) \underset{\{service\}}{\bowtie} Sink\right] \end{gathered}$$

Here, $PSource$ is a $Source$ of positive customers. Its action is relabelled to $arrival$. Similarly, $NSource$ produces negative customers using the relabelled action $kill$. If positive customer blocking, tail-safe killing mode or multiple processors $c \neq 1$ are desired, some queue definitions need to be altered and special cases inserted for a range of queue lengths $Q_1 \ldots Q_c$. These expressions would not be additionally enlightening and are therefore omitted here here for brevity.

5.3 Solving for the steady-state of the network

In seeking the joint steady-state probability distribution of all the objects within a network, we quickly run into the problem of state space explosion when using tools such as the PEPA workbench [GilHil94] on the PEPA model. To overcome this computational limitation, we propose an iterative algorithm that approximates the network behaviour by solving individual components within the network with subsequent combining of the results. In the following discussion we will initially consider queues with only one server (*i.e.* $c = 1$) and no modulation. We will then generalise the algorithm to allow a full specification for multi-servers to accommodate modulation.

5.3.1 Departure Traffic for systems without modulation

The fundamental idea behind the iterative algorithm is to maintain an approximation of the departure traffic originating from every object or node within the system. These traffic estimates are then used to gain more accurate steady-state solutions for each queue, which, in turn, improves the departure traffic approximations. To enable subsequent solution using the methods described in chapter 3, traffic descriptions should be of a form compatible to the arrival processes that are allowed in our queueing model. This means that the most general form we should allow (in the absence of modulation, for now), is that of a Poisson stream with geometric batches. A full characterisation is given by the arrival rate and batch distribution parameters λ and θ that are already familiar from the arrival parameters in section 2. Commonly, the use of just one such stream gives a coarse traffic approximation. When this is the case, we can generally improve the fit by using multiple, superimposed processes given by a set of parameters λ_k and θ_k.

Sinks cannot have departures such that all traffic is either originated by a Source or is as a result of a service stream at a Queue. We initially propose two simple methods of approximation. The results gained from these are then combined in a method of higher accuracy. For traffic emerging from a Source, no approximations is necessary as the Source definition is already in terms of rate and batch parameters. The challenge

therefore lies in finding sufficiently accurate approximations for a Queue departure process. Here, the effective rate and batch parameters are affected by the service rate μ and service batch parameter ϕ as well as its steady-state state occupation probability distribution π_j for $0 \leq j \leq L$. The state occupation probabilities are important because a queue with a very high processing rate will not have a similarly high departure rate if it is idle most of the time.

5.3.2 Poisson approximation

The simplest Markovian traffic approximation involves matching the mean rate of a departure stream by a simple Poisson distribution. Our traffic representation is then given by a single rate parameter t_R for an exponential distribution that approximates the inter-arrival time distribution of successive customers. For our approximation to be accurate it is necessary that the matched stream has the same throughput as the actual departure stream. To calculate queue throughput, we note that the mean departure batch size at queue level j is given by

$$\begin{aligned}
\overline{s}_j &= \textstyle\sum_{s=1}^{j-1} s(1-\phi)\phi^{s-1} + j\sum_{s=j}^{\infty}(1-\phi)\phi^{s-1} \\
&= \tfrac{1-\phi^j(j-j\phi+\phi)}{1-\phi} + j\phi^{j-1} \\
&= \tfrac{1-\phi^j-j(1-\phi)\phi^{j-1}}{1-\phi} + j\phi^{j-1} \\
&= \tfrac{1-\phi^j}{1-\phi} - j\phi^{j-1} + j\phi^{j-1} \\
&= \tfrac{1-\phi^j}{1-\phi}
\end{aligned}$$

Using this expression, the throughput T for a constant rate server is as follows

$$\begin{aligned}
T &= \mu\textstyle\sum_{j=1}^{L}\pi_j\overline{s}_j \\
&= \tfrac{\mu}{1-\phi}\textstyle\sum_{j=1}^{L}\pi_j(1-\phi^j) \\
&= \tfrac{\mu}{1-\phi}\left(\textstyle\sum_{j=1}^{L}\pi_j - \sum_{j=1}^{L}\pi_j\phi^j\right) \\
&= \tfrac{\mu}{1-\phi}\left(1-\pi_0-\textstyle\sum_{j=1}^{L}\pi_j\phi^j\right) \\
&= \tfrac{\mu}{1-\phi}\left(1-\textstyle\sum_{j=0}^{L}\pi_j\phi^j\right)
\end{aligned}$$

The throughput of a Poisson stream per unit time is equal to its parameter. Hence choosing $t_R = T$ gives a description of traffic that matches the observed departure process of the queue.

5.3.3 One batch approximation

We augment the Poisson traffic approximation to allow for the presence of batches in our second simple approximation method. As before we use t_R to represent the approximate traffic rate, but in addition we use t_B as a batch parameter of the departing traffic. The approximation we propose involves choosing the batch parameter t_B that yields the same average batch size as the steady-state solution of our queue. Given this batch parameter, we then find the rate parameter t_R that matches the throughput.

Using the average batch size at queue length j found earlier, we observe that

$$\overline{s} = \frac{\sum_{j=1}^{L} \overline{s}_j \pi_j}{1-\pi_0} = \frac{1-\sum_{j=0}^{L} \pi_j \phi^j}{(1-\pi_0)(1-\phi)}$$

gives the overall average output batch size of the queue when weighted on its state occupation probabilities. The mean batch size of a geometric stream with batch parameter t_B is given by

$$\sum_{s=1}^{\infty} s(1-t_B)t_B^{s-1} = \frac{1}{1-t_B}$$

Matching mean batch sizes gives

$$\begin{aligned} \tfrac{1}{1-t_B} &= \overline{s} \\ t_B &= 1-\tfrac{1}{\overline{s}} \end{aligned}$$

The value of the other variable, t_R, is now chosen to again match the throughput T, *i.e.*

$$t_R \frac{1}{1-t_B} = T$$

or

$$t_R = \frac{T}{\overline{s}}$$

5.3.4 Utilisation weighted approximation

Comparing the queue length distributions that result from using the two simple traffic descriptions given above with the actual one provided by a simulator (given in chapter 5.6), indicates that both approximations to link traffic result in systematic errors,

which provide upper and lower bounds. We have found that while the Poisson approximation underestimates the mean queue length, the one batch method overestimates it. This observation is consistent with the conjecture in [KouAlm88]. This claims that GE distributions give pessimistic performance estimates whereas Exponentials give optimistic estimates. In [HarTho+02] we used a mixture of the two approximation methods that improves on the accuracy given by either simple method individually. The idea is to use two independent, superimposed traffic streams to represent the departure traffic. As before, the Poisson stream is given by a rate parameter λ_1 and batch parameter 0 whereas the batched component is represented by λ_2 and θ_2. Two of the constraints we apply to these parameters are that the constituent batched stream has the same mean batch size as the departures from the queue and that the combined throughput is consistent with the queue.

A third constraint is needed to make the choice of parameters unique. A particular choice that has been found to give encouraging results [HarTho+02] is to make the contribution of the Poisson stream to the total traffic proportional to the utilisation $\rho = 1 - \pi_0$ of the queue. The rationale is that a queue that is lightly loaded is more Poisson-like, with bursty departures being less frequent. Consequently, we propose to use the following set of constraints for the weighted approximation.

$$\begin{aligned} \theta_2 &= 1 - \tfrac{1}{\bar{s}} \\ T &= \lambda_1 + \tfrac{\lambda_2}{1-\theta_2} \\ \rho\lambda_1 &= (1-\rho)\tfrac{\lambda_2}{1-\theta_2} \end{aligned}$$

or, equivalently

$$\begin{aligned} \theta_2 &= 1 - \tfrac{1}{\bar{s}} \\ \lambda_1 &= T(1-\rho) \\ \lambda_2 &= \tfrac{T\rho}{\bar{s}} \end{aligned}$$

5.4 Iterative algorithm for the network solution

We now describe how these approximate traffic descriptions can be used in an iterative algorithm to find the steady-state probability distribution of each queue within a network. At every iterative step, we feed our current link traffic approximations into the

arrival processes of the relevant queues and solve each single-queue system in isolation, using the methods developed in chapter 3. The resulting steady-state distribution is then used to derive an improved departure traffic representation that is then available for subsequent, downstream queue solutions.

5.4.1 Initialisation

At the start of the algorithm when neither steady-state solutions nor approximate traffic descriptions are known, we initialise using *traffic equations* (see *e.g.* [HarPat92, Mit87]). When solved, these equations give the mean arrival rates we expect at each queue in the absence of losses that might occur through blocking. As such, the solutions give a rough estimate of the link traffic present in a system with batched traffic. However such approximations are adequate for the purpose of initialising the traffic estimates. Using our notation, traffic equations are given by the linear system

$$\underline{\mathbf{Y}}(\overrightarrow{\mathbf{t}} + \overrightarrow{\mathbf{s}}^{in}) = (\overrightarrow{\mathbf{t}} + \overrightarrow{\mathbf{s}}^{out})^T$$

where $\underline{\mathbf{Y}}$ is the routing matrix, $\overrightarrow{\mathbf{t}}$ the sought vector of the incoming internal traffic to each queue, $\overrightarrow{\mathbf{s}}^{in}$ the vector of the (known) traffic rates originating from sources outside of the network and finally $\overrightarrow{\mathbf{s}}^{out}$ gives the mean traffic rates with which traffic arrives at Sinks. The $\overrightarrow{\mathbf{t}}$-vector has zero entries at positions corresponding to sources and Sinks and $\overrightarrow{\mathbf{s}}^{in}$ has zero entries at positions corresponding to queues and Sinks. For $\overrightarrow{\mathbf{s}}^{out}$, we have zeros for queues and sources.

For our example system from before, we therefore have

$$\overrightarrow{\mathbf{t}} = (0, t_1, t_2, t_3, 0)^T \quad , \quad \overrightarrow{\mathbf{s}}^{in} = (s_1^{in}, 0, 0, 0, 0)^T \quad and \quad \overrightarrow{\mathbf{s}}^{out} = (0, 0, 0, 0, s_5^{out})^T$$

and the linear system is

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{6} & 0 & 0 & 0 \\ 1 & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{4} & 0 \end{pmatrix} \begin{pmatrix} s_1^{in} \\ t_1 \\ t_2 \\ t_3 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & t_1 & t_2 & t_3 & s_5^{out} \end{pmatrix}$$

The unknowns are given by the t_i and s_i^{out}. For our purposes it is not necessary to find any values contained within $\overrightarrow{\mathbf{s}}^{out}$, because this vector does not correspond to any traffic that is internal to the network and is not needed for traffic approximation purposes. We can therefore discard those equations with s_i^{out} on the right hand side.

The routing matrix $\underline{\mathbf{Y}}$ has a zero-row for each Source because traffic cannot be directed there. This results in an equation of the form $0 = 0$ which does not give any constraint on variables. In addition, the zero-column associated with the Sink has also no bearing on the solution and consequently these equations can also be dropped. We are therefore left with the following reduced system

$$\begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{4} \\ 0 & \frac{1}{6} & 0 & 0 \\ 1 & \frac{1}{2} & 1 & 0 \end{pmatrix} \begin{pmatrix} s_1^{in} \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} t_1 & t_2 & t_3 \end{pmatrix}$$

When solved, the values t_i give the desired average rates into each queue component. For example, when we have an external arrival rate $s^{in} = 1$, the solution is

$$(t_1, t_2, t_3) = \left(\frac{1}{2}, \frac{1}{12}, \frac{4}{3}\right)$$

At this point, since no information on the batch behaviour for the departure traffic is available, we will ascribe the batch parameter 0 to each, using the solution of the traffic equation as the rate parameter. The effect of these considerations is that initially all network traffic is represented as unbatched Poisson.

5.4.2 Construction of the arrival process

Depending on the routing matrix, traffic streams between any two queues can be *thinned* (probabilistically split) and joined in several different ways. The challenge lies in how to use the routing matrix $\underline{\mathbf{Y}}$ in combination with the approximate departure traffic rates to derive an arrival process that can be fed into each queue's isolated model.

Thinning is easily achieved by adjusting the traffic rate according to the routing probability. A batched stream with rate parameter λ and batch parameter θ, when routed

with probability p to its target, arrives as a batched stream with rate $p\lambda$ and unchanged batch parameter θ.

The combining of a number of (possibly thinned) arriving streams into the arrival process can be undertaken in two ways: by superposition or aggregation. Superposition is the more accurate, but computationally more expensive method. This involves superimposing the arriving traffic, allocating as many arrival streams as necessary in the solution approach of chapter 3. The number of arrival streams is not necessarily equal to the number of traffic sources to a particular queue. We also need to take into account the traffic approximation method used. For the weighted approximation method, departure traffic is represented by two superimposed processes, so that each adds not one, but two arrival processes. Traffic approximated by the weighted model consists of one Poisson and one batched stream, which allows us to combine all the pure Poisson streams into one whose rate is the sum of the rates of its constituent parts. This approximation method is therefore only marginally more expensive than the one batch approximation.

Each additional independently batched stream adds to the computational cost of a queue in two ways. During formulation of its balance equations more elimination steps are required (see Appendix A), and the solution involves more eigenvalues. It can therefore be advantageous to employ an aggregation method that approximates the joint behaviour of all arriving traffic descriptions using just one arrival stream. This method is comparable with the results of applying *Maximum Entropy* (ME) methods [Jay57], developed further in a queueing context by Kouvatsos (*e.g.*[KouAwa03]), where a GE-distribution is also used to approximate revised arrival processes at each queue of the network. This network includes blocking and contains multiple classes of customers, something that our model cannot accommodate without extension.

5.4.3 Algorithm

The outline of the algorithm we use is the following:

• Define the queueing network of Queue objects $Q = \{Q_1, \ldots, Q_n\}$, $\underline{\mathbf{Y}}$ and rate parameters

• Initialise the departure traffic $T = \{t_1, \ldots, t_n\}$ from Q, $\underline{\mathbf{Y}}$ and the external arrival rates using traffic equations

do

for $i = 1$ to n

• From $\underline{\mathbf{Y}}$ and T, Construct arrival process at Q_i

• Solve Q_i at steady-state, using previously found arrival process, using methods from chapter 3 and Appendix A

• Find new approximation for the departure traffic t_i

end

until converged

Convergence is determined by monitoring some measure that approaches zero as the system settles. A wide variety can be used here. The simplest being the maximum change in mean queue lengths over all queues or the maximum change in traffic approximation parameters found during the last iteration. In our calculations we have chosen to use the former.

5.4.4 Example Solution

We now present a sample solution to the simple network shown earlier. So far, only the routing probabilities have been specified. To uniquely determine the network behaviour, we additionally select buffer sizes, service rates and batch size parameters of queues as well as the traffic properties of the external source. In the following, we set all buffer sizes to $L = 30$. The external source is Poisson with a rate of $\mu = \frac{3}{2}$. Queue

service characteristics are, in order,

$$Q_1\text{:}\quad \mu = \tfrac{1}{3} \quad \phi = \tfrac{1}{4}$$
$$Q_2\text{:}\quad \mu = \tfrac{1}{4} \quad \phi = \tfrac{1}{10}$$
$$Q_3\text{:}\quad \mu = 1 \quad \phi = \tfrac{1}{2}$$

To give an indication of the accuracy of approximations using the various traffic representation methods, we include the result of an expensive simulation process, whose methodology is described in chapter 5.6, and which is run over 10^9 network state transitions. Figures 5.4, 5.5 and 5.6 show the steady-state queue length distributions achieved by the various methods for our 3 queues. For all three queues the weighted

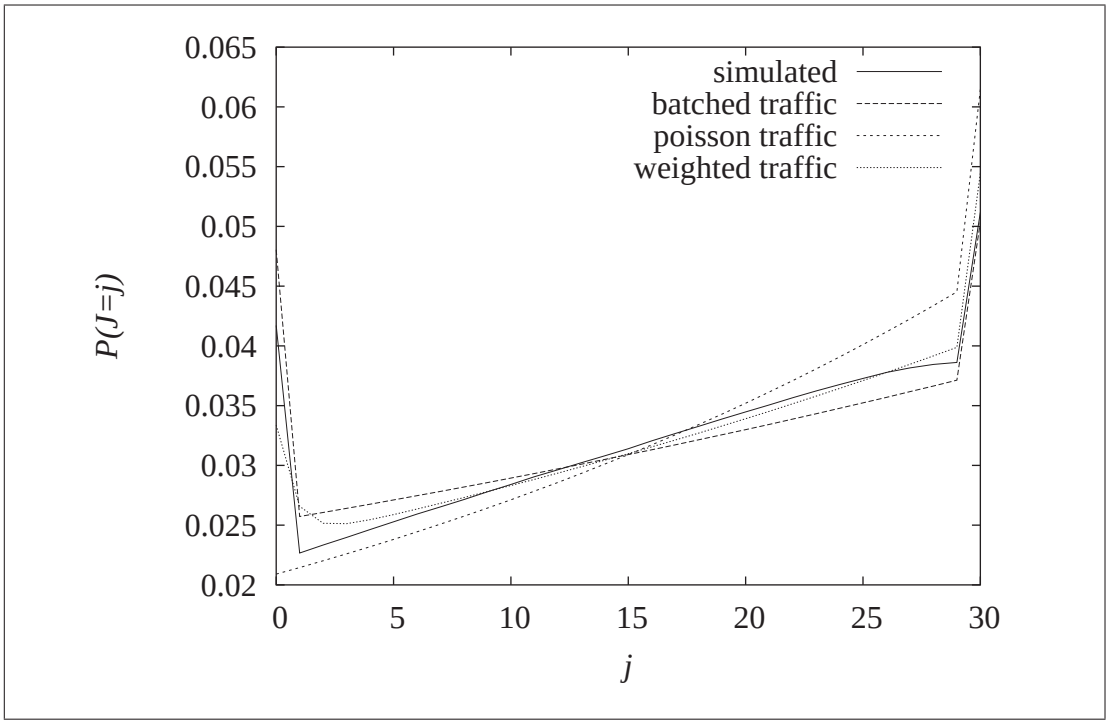


Figure 5.4: Queue length distribution at steady-state for Q_1

approximation delivers the best accuracy compared to the other two methods. When one overestimates the mean queue lengths, the other tends to underestimate it. The mixture of Poisson and geometrically batched traffic approximations achieves a superior solution. However, there is still a significant discrepancy between the actual probabilities and the approximated ones. We suggest that this is largely due to approximate traffic shapes and correlation. The two simulators we present in sections 5.7 and 5.8 will shed some light on which is more significant.

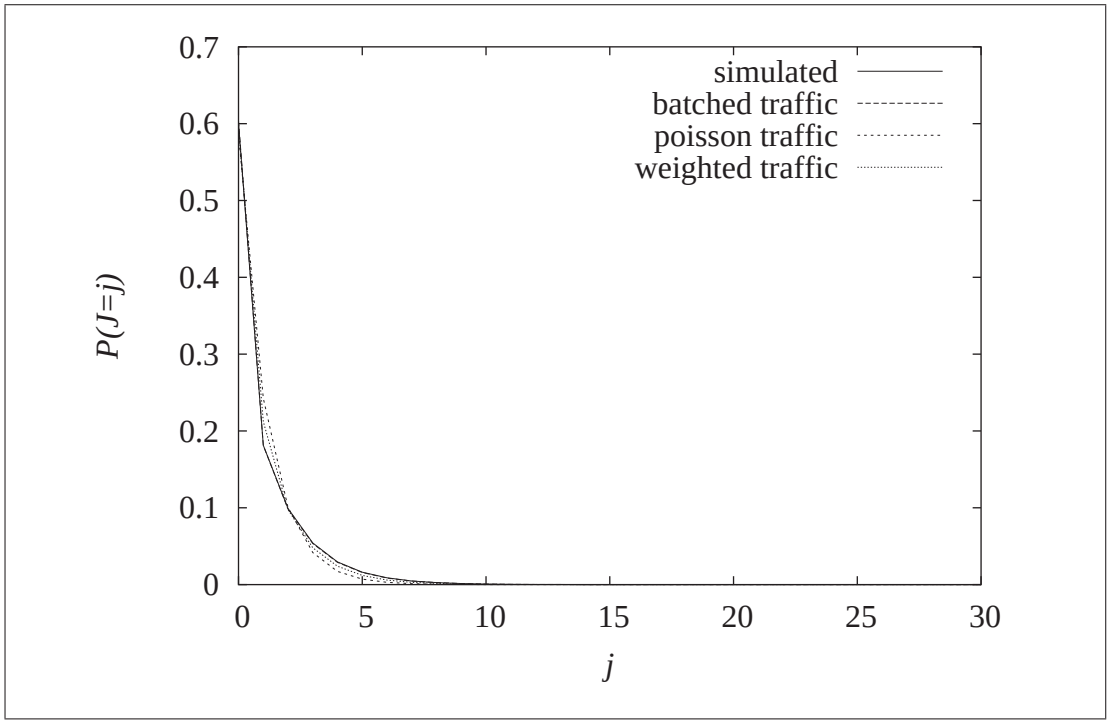


Figure 5.5: Queue length distribution at steady-state for Q_2

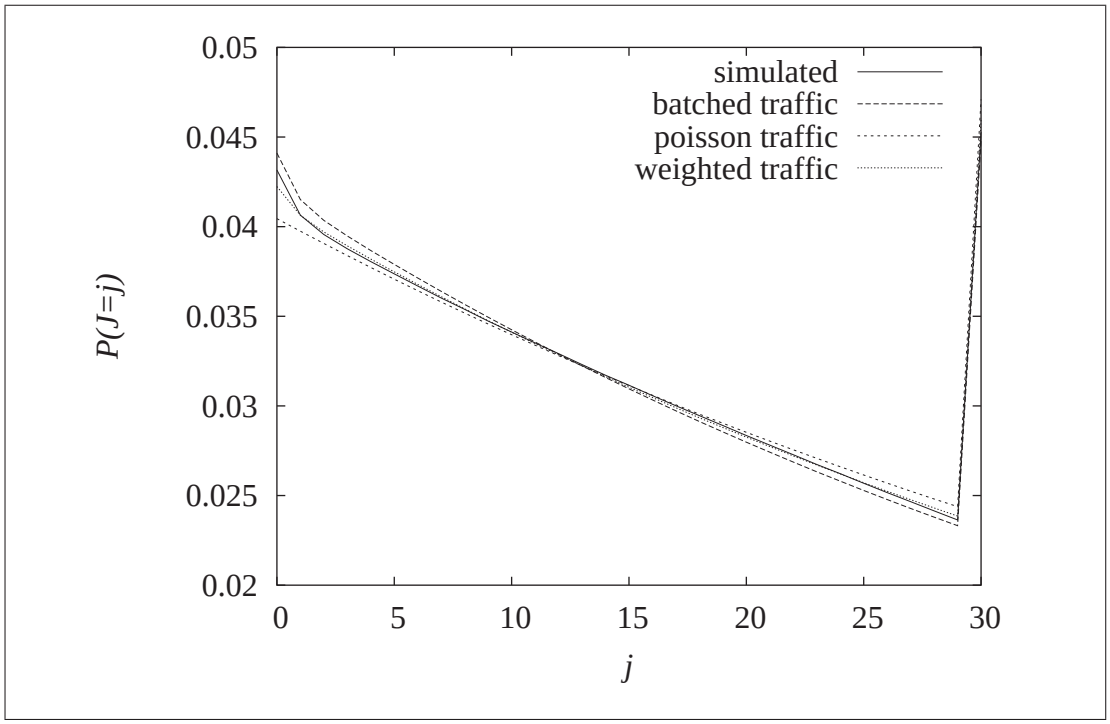


Figure 5.6: Queue length distribution at steady-state for Q_3

5.5 Modulated Multi-servers

In this section, we generalise the expressions provided for mean batch sizes and for traffic rates that were derived above. The incorporation of modulation is simply accomplished by using a vector expression, with each component of the vector corresponding to a modulation phase. The presence of a multi-server complicates matters slightly because service batches are now truncated to level $c-1$. That is when $c > 1$ the queue cannot empty in one transition, irrespective of the size of the service batch. Therefore we need to use the matrix $\underline{\mathbf{C}}_j$ from chapter 2.5, where it was defined as a diagonal matrix with diagonal element $(\underline{\mathbf{C}}_j)_{m,m} = \min(j, c_m)\mu_m$, while c_i are the number of processors active in modulation state i, $1 \leq i \leq N$.

We write

$$\overrightarrow{\overline{\mathbf{s}}_\mathbf{j}} = (\overline{s}_{j,1}, \ldots, \overline{s}_{j,N})$$

where

$$\overline{s}_{j,n} = \begin{cases} 1 & \text{If} \quad j \leq c_n \\ \sum_{s=1}^{j-c_n} s(1-\phi)\phi^{s-1} + (j-c_n+1)\sum_{s=j-c_n+1}^{\infty}(1-\phi)\phi^{s-1} & \text{otherwise} \end{cases}$$

The expression for the truncated geometric batches simplifies as follows

$$\begin{aligned} &\textstyle\sum_{s=1}^{j-c_n} s(1-\phi)\phi^{s-1} + (j-c_n+1)\sum_{s=j-c_n+1}^{\infty}(1-\phi)\phi^{s-1} \\ &= \tfrac{1-\phi^{j-c_n}}{1-\phi} + (c_n - j)\phi^{j-c_n} + (j-c_n+1)\phi^{j-c_n} \\ &= \tfrac{1-\phi^{j-c_n}}{1-\phi} + \phi^{j-c_n} \end{aligned}$$

Therefore,

$$\overline{s}_{j,n} = \begin{cases} 1 & \text{If} \quad j \leq c_n \\ \frac{1-\phi^{j-c_n}}{1-\phi} + \phi^{j-c_n} & \text{otherwise} \end{cases}$$

The modulated, multi-server measures for throughput T and mean departing batch size $\overline{s}$ are given by

$$T = \sum_{j=1}^{L} \overrightarrow{\mathbf{v}}_j^T \underline{\mathbf{C}}_j \overrightarrow{\overline{\mathbf{s}}_\mathbf{j}}$$

$$\overline{s} = \frac{\sum_{j=1}^{L} \overrightarrow{\mathbf{v}}_j^T \underline{\mathbf{C}}_j \overrightarrow{\overline{\mathbf{s}}_\mathbf{j}}}{\sum_{j=1}^{L} \overrightarrow{\mathbf{v}}_j^T \underline{\mathbf{C}}_j \overrightarrow{\mathbf{e}}}$$

These expressions user the vector equivalent of the scalar terms in sections 5.3.2-5.3.4. We derive average measures for each modulation state, weighted on being in the states, to obtain overall quantities that are valid for the queue traffic.

By also using $\rho = 1 - \overrightarrow{\mathbf{v}}_0^T \overrightarrow{\mathbf{e}}$, these expressions allow us to give Poisson, one batch and weighted departure traffic approximations. Note that whereas the departure process is modulated, the traffic approximation is not. That is we are averaging the departure behaviour over all modulation states and representing it by a single, unmodulated stream.

5.5.1 Modulated Traffic

We can aim to capture more traffic features by retaining the modulation behaviour in the traffic description. The trade-off here is that we will subsequently need to merge the multiple, independently modulated traffic streams that are fed into a queue with the modulated system of the queue itself. The merging process yields a system with the number of modulation states given by the product of the number of all constituent modulation states. For unmodulated traffic streams, with $N = 1$, this is of course never a problem.

Modulated traffic is modelled by representing the quantities T and $\overline{s}$ that were scalar before, as vectors, $\overrightarrow{\mathbf{T}}$ and $\overrightarrow{\overline{\mathbf{s}}}$. Now,

$$\overrightarrow{\overline{\mathbf{s}}} = (\overline{s}_1, \ldots, \overline{s}_N)$$

$$\overline{s}_n = \frac{\sum_{j=1}^{L} \overrightarrow{\mathbf{v}}_j^T \underline{\mathbf{C}}_j \overrightarrow{\mathbf{e}}_n \overline{s}_{j,n}}{\sum_{j=1}^{L} \overrightarrow{\mathbf{v}}_j^T \underline{\mathbf{C}}_j \overrightarrow{\mathbf{e}}_n}$$

and

$$\overrightarrow{\mathbf{T}} = \sum_{j=1}^{L} \overrightarrow{\mathbf{v}}_j^T \underline{\mathbf{C}}_j \underline{\mathbf{diag}}(\overrightarrow{\overline{\mathbf{s}}_\mathbf{j}})$$

where $\underline{\mathbf{diag}}(\overrightarrow{\mathbf{x}})$ is the diagonal matrix with the components of $\overrightarrow{\mathbf{x}}$ on the diagonal.

The modulated Poisson traffic approximation is now given by

$$\overrightarrow{\mathbf{t_R}} = \overrightarrow{\mathbf{T}}$$

$$\overrightarrow{\mathbf{t_B}} = \overrightarrow{\mathbf{0}}$$

whereas the one batch approximation has the parameters

$$\overrightarrow{\mathbf{t_R}} = \overrightarrow{\mathbf{T}}^T \underline{\mathbf{diag}}(\overrightarrow{\mathbf{s}})^{-1}$$

$$\overrightarrow{\mathbf{t_B}} = \overrightarrow{\mathbf{e}} - \overrightarrow{\mathbf{e}}^T \underline{\mathbf{diag}}(\overrightarrow{\mathbf{s}})^{-1}$$

Finally, the weighted approximation, with weighting vector $\overrightarrow{\rho}$ gives

$$\begin{aligned}
\overrightarrow{\lambda}_1 &= \overrightarrow{\mathbf{T}}^T(\underline{\mathbf{I}} - \underline{\mathbf{diag}}(\overrightarrow{\rho})) \\
\overrightarrow{\lambda}_2 &= \overrightarrow{\mathbf{T}}^T \underline{\mathbf{diag}}(\overrightarrow{\rho}) \underline{\mathbf{diag}}(\overrightarrow{\mathbf{s}})^{-1} \\
\overrightarrow{\theta}_1 &= \overrightarrow{\mathbf{0}} \\
\overrightarrow{\theta}_2 &= \overrightarrow{\mathbf{e}} - \overrightarrow{\mathbf{e}}^T \underline{\mathbf{diag}}(\overrightarrow{\mathbf{s}})^{-1}
\end{aligned}$$

5.6 Network Simulators

At several stages in the network solutions presented above approximations were employed to keep the solution process simple and efficient. However, it is conceivable that this approximation effort has altered the properties of the system in such a way, that the steady-state distribution results are unreliable and do not reflect actual behaviour.

To identify cases that lend themselves quite well to our approximation methodology as well as avoiding those that give unacceptably high errors, we will present two simulators that allow us to identify where (if any) in the approximation chain errors are introduced. The two simulators themselves differ in accuracy and execution speed. As is to be expected, the slower one gives the more faithful solution, whereas the faster one trades off some of the accuracy against reduced computational effort.

Whenever the solution process relying on the departure-traffic approximation is known to be inaccurate, we can use one of the simulators to give accurate steady-state data before proceeding to the sojourn time calculations. This enables us to provide accurate results in all situations.

5.7 Job-driven simulator

Our aim for our first simulator is to give the most accurate reproduction of queue behaviour, at the same time collecting data for the wide array of performance charac-

teristics that we are interested in. We use this information to judge the accuracy of the various traffic matching approaches.

The specific measures we gather are state occupation probabilities, blocking and/or loss probabilities, inter-arrival delay distributions conditioned on the queue length of the queue at which an arrival occurs, as well as age distributions[1] of customers upon arrival, service and killing at a given queue. Whereas the first two measures would be available in a simulation with anonymous customers, *i.e.* where only the number of customers at some stage within the network is monitored, the last few measures require that customers be individually tagged with a 'time of birth', to enable us to process the timing information we need to sample from. This simulator is therefore called the *Job-driven* simulator, to underline that individual customers (jobs) are being simulated.

Each queue is allocated a FIFO (FCFS) buffer for each queue where incoming customers queue before processing. We employ the queue model from chapter 2, so all actions within the network follow at exponentially distributed intervals. This is true for external source arrivals as well as queue departures and modulation state changes. With all transitions therefore memoryless, we can track the progress of the network by way of a global clock that is advanced by the delay caused by the next network activity. There is no subsequent need to use delays that depend on the previously performed action because of the *memoryless property*.

5.7.1 Initial population of networks

We allow for the initial condition of the network to be specified. Essentially, it specifies the modulation phases of the various queues and the numbers of customers in their waiting rooms. When applying steady-state analysis on open networks, this initial condition does not matter. We provide the facility to specify the initial condition so that the simulator can be used for a wider range of problems including those of closed networks and transient analysis. In addition, judicious choices for the initial number of

[1] we define the age of a customer as the length of time it has spent within the network. This measure is only meaningful for open networks.

customers in a queue can speed up simulation convergence dramatically, by assigning a number of customers equal to the mean expected queue length derived from *e.g.* traffic equations.

5.7.2 Simulation details

Each node in the network ($Source$, $Sink$ or $Queue$) is designed in such a way that it can, independently from other nodes, carry out its active actions $modulation$ and $service$, when triggered to do so by a governing process, the *scheduler*. The scheduler itself is responsible for simulating the race conditions between the possible actions: A Process's modulation or service should only then be triggered by the scheduler when it may be that process turn were it to perform such an action. To determine which process will be carrying out the next action, the Scheduler queries every object in the network for the delay of its next action[2] jointly with the type (modulation or service) of the said action. The scheduler then picks the $(object, action, delay)$ tuple with the smallest of all the returned delays as the next event. Other events that have not been picked are discarded. If general interval distributions were allowed, it would be necessary to keep track of these events or at least condition subsequent scheduler queries on not having been picked before. Fortunately there is no need for this because, as remarked earlier, all inter-transition intervals are exponentially distributed and hence memoryless. The simulation algorithm is therefore (in outline) as follows

1. Initialise time $t = 0$
2. Query all objects
3. Determine the next event (object $\hat{o}$, action $\hat{a}$, delay $\hat{d}$)
4. Increment $t := t + \hat{d}$
5. Trigger the appropriate action at Object $\hat{o}$
6. Return to 2 until termination criterion is met

[2] if there is no active action, ∞ is returned as delay

5.7.3 Termination criterion

It frequently is a problem to determine when exactly a steady-state experiment has 'sufficiently' settled down so as to make the results accurate enough for its intended purposes. The criterion we propose calculates the maximum difference of some performance measure between regular intervals and exits when this maximum is less than some pre-determined threshold. A common measure that works well and is simple to calculate are mean queue lengths. When this is used, the maximum change over the course of 10 million (say) state transitions in any given mean queue length for all queue objects is calculated and if smaller than some ϵ, approximate steady-state has been reached. To avoid both very short and very long simulation runs, we specify, in addition, minimum and maximum run times (in number of state transitions effectuated), so that upper bounds on execution time can be enforced.

5.8 Pdf-driven simulator

Whereas the job-driven simulator aimed to precisely follow the operation of the actual system, we propose a second simulator that will allow us to judge at which approximation stage the solution breaks down (if it does). At two stages in the analytical matching process, we simplify the departure traffic at each queue.

The first approximation happens when we represent the departure traffic using (possibly multiple) exponential delays with geometrically distributed batch sizes. There is no guarantee that such a representation will successfully capture all relevant traffic aspects. In particular, departure batch sizes from queues of finite length are invariably limited to the size of the waiting room + 1, as opposed to geometric distributions which are unbounded. This is a principal danger when using such generalised exponential (GE) distributions, so we must be very specific about its effects.

The mechanics of the solution process, where each queue is solved in turn, gives the other source of discrepancies between the actual model traffic and the matched traffic. By solving queues in isolation and subsequently updating the current estimate

for the departure traffic, all information about the correlation between two queues is lost. To illustrate an instance of this behaviour, we consider an example taken from [ThoZat03], the three queue feedback network in figure 5.7.

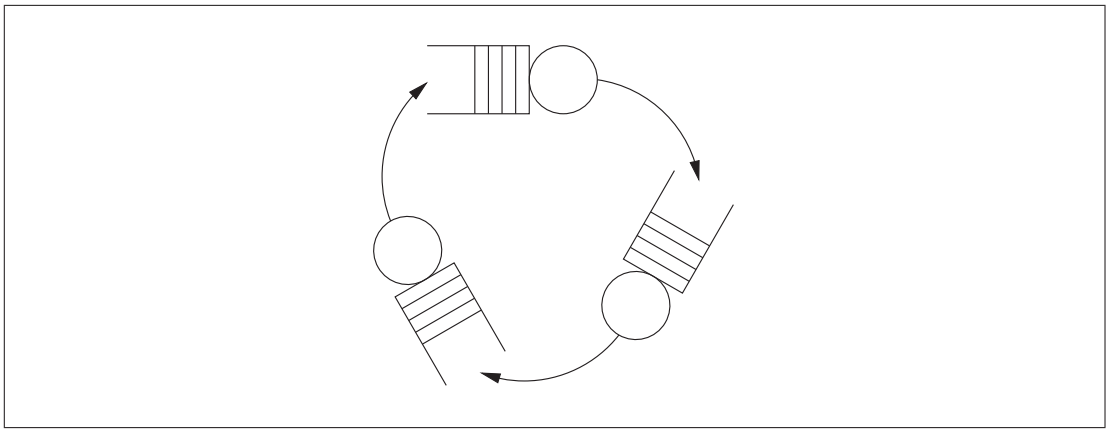


Figure 5.7: Three queue feedback network

In [ThoZat03] we constructed a network population of 60 and gave each of the three queues identical service characteristics: service rate $\mu = 1$, service batch parameter $\phi = 9/10$ and buffer size $L = 60$. In addition to recording queue length distributions, the simulation was instrumented to record the inter-arrival distributions and the sizes of batches arriving at a queue for each given queue length of the target queue.

From the nature of the feedback network, we can expect strong correlation to exist between the queue lengths of the three queues. At its simplest, the more customers one particular queue contains, the fewer are left over for the others in this closed system. As a consequence, the arrival process at the queue is effectively throttled at large queue lengths, and reaches zero when the queue contains all 60 customers. The correlated behaviour therefore gives rise to a queue length dependent arrival process. Figure 5.8 shows the inter-arrival and arrival batch size distributions that a queue in this system experiences for different queue lengths. At low queue lengths, the inter-arrival delay and arrival batch distributions behave as would be expected from an exponential rate and geometric batch size. With increasing queue length, the inter-arrival time distribution flattens and develops a local maximum, whereas the batch distributions become visibly truncated at the batch size that would cause all customers within the system to

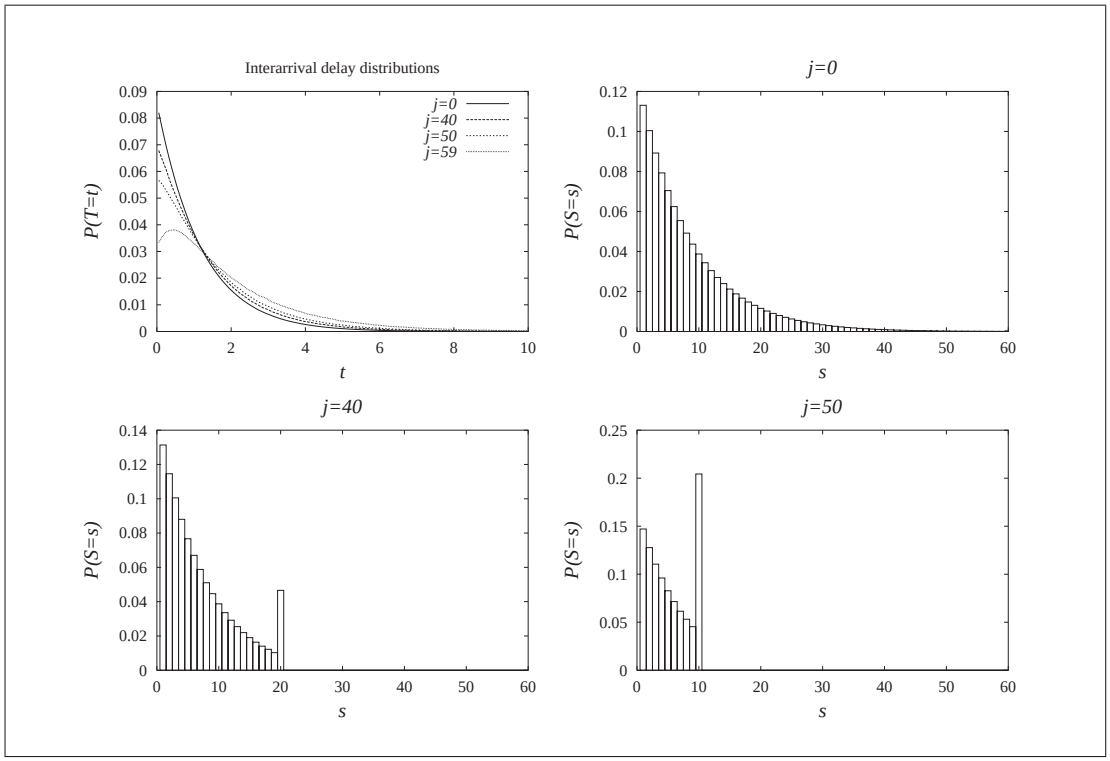


Figure 5.8: Inter-arrival and Batch Distributions at selected queue lengths

be at this queue.

From inspection of the mean and standard coefficient of variation (SCV) of the arrival rates and arrival batch distributions in figure 5.9, it is clear that a simple rate/batch parameter combination cannot accurately describe the arrival process at all queue lengths. Correlated intra-queue traffic is the cause of this behaviour. The three queue feedback network is designed to be an extreme example, to illustrate potential problems in our solution process. Queues are in a closed network, within a tight loop, with large average batch sizes, allowing rapid queue length changes between the three queues,

By changing the network into an open network, this strong correlation between queue length and arrival process (and hence departure process from other queues) is significantly weakened, but it does not disappear entirely.

To investigate which of the two potentially error-inducing approximations described above cause inaccuracies in the steady-state results, we propose a simulator that introduces only one of these approximations, while being entirely accurate for the other. The *pdf-based* simulator allows arbitrarily-shaped inter-departure time and batch dis-

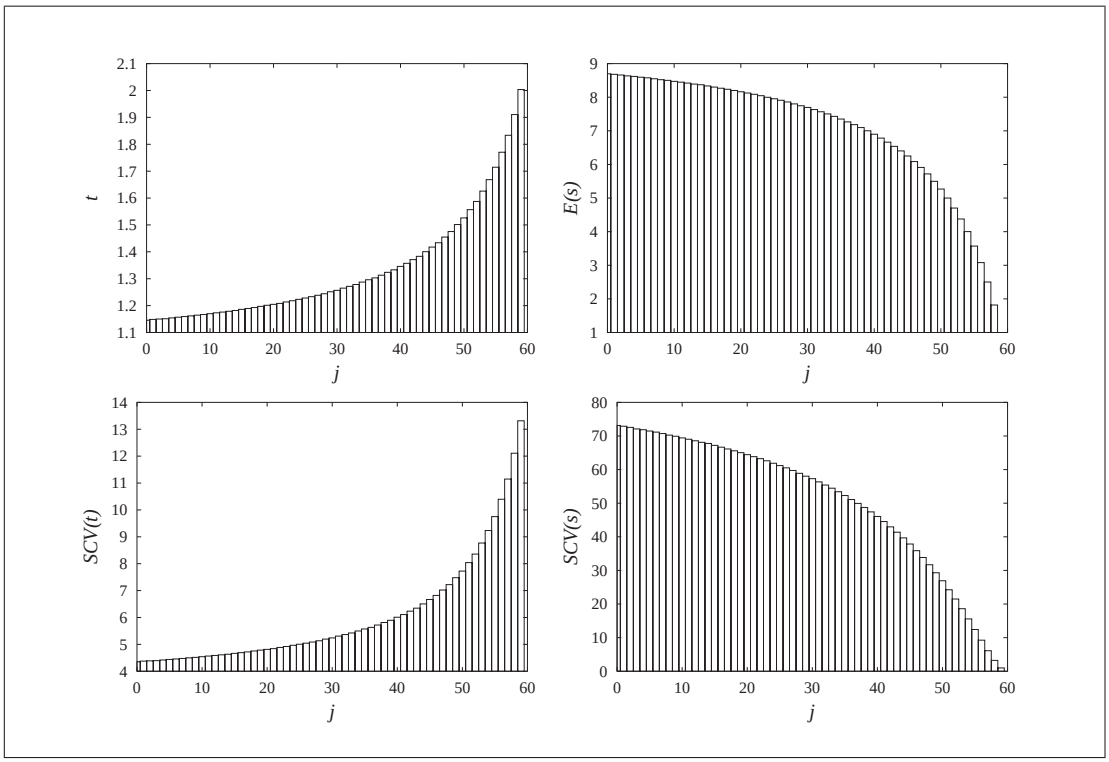


Figure 5.9: Mean and SCV of Inter-arrival and Batch Distributions

tributions. At the same time, it behaves according to the matching methodology in that it assumes independence between queues by simulating the queues separately, only passing the (empirical) density functions of the departure traffic between the queues. Comparisons of the steady-state distributions of our three methods (job-driven simulator, pdf-driven simulator and analytical traffic matching), then allow us to determine whether the independence assumption or the lack of descriptiveness of link traffic are primarily responsible for inaccuracies.

5.8.1 Parallel implementation

The simulated systems can be very large, so we seek to exploit a natural method of parallelising the simulation. Due to the nature of the pdf-simulation approach, which (just as the matching approach) solves each queue in turn, there might be a simple form of parallelism that can be exploited, depending on the network topology. Each queue within the network is assigned a separate processor that occupies itself exclusively

with finding the equilibrium SOPs and the departure characteristics. Periodically, all processors stop and write out the current estimates of their departure traffic characteristics and update their incoming traffic approximations that result from the other processors simulations. The inter-processor communication is achieved via a shared file-space, where files contain traffic and queue length distribution information.

5.9 Accuracy comparisons

We start our comparison between the pdf-based simulator and the various matching methodologies by considering a *feed forward* network, *i.e.* a network without any loops. It consists of two single-server, unmodulated queues. The first queue has external Poisson arrivals with rate $18/10$ and processes at rate 1, with batch parameter $1/2$. Departures from the first queue are then routed to the second queue, whose service process also has rate 1 and batch parameter $1/2$. Upon leaving the second queue, customers exit the system. The network is pictured in figure 5.10.

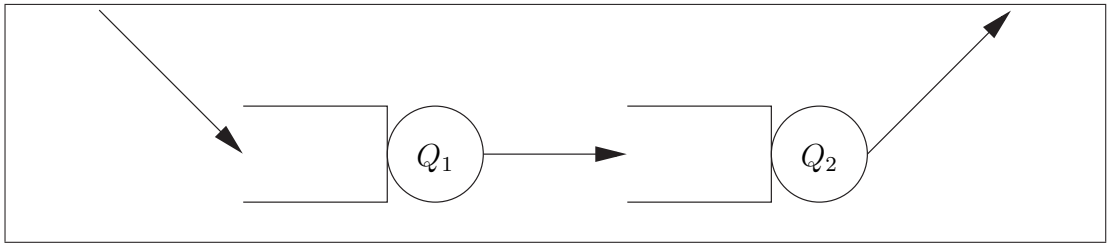


Figure 5.10: Topology of the feed forward example network.

The equilibrium SOPs of Q1 will be accurate for all approximation approaches because the incoming traffic description (*i.e.* the Poisson stream with rate $18/10$), is known exactly. We will therefore only consider Q2, whose incoming traffic is only known approximately. Figure 5.11 shows the probability distributions resulting from the various traffic approximations. As expected, the pdf-based simulator, being the theoretical best traffic approximation method, is superior to all other matching methodologies. Among the latter, the utilisation weighted approximation fares best, and the Poisson traffic approximation is worst. Figure 5.12 plots the errors against the accurate SOPs, which

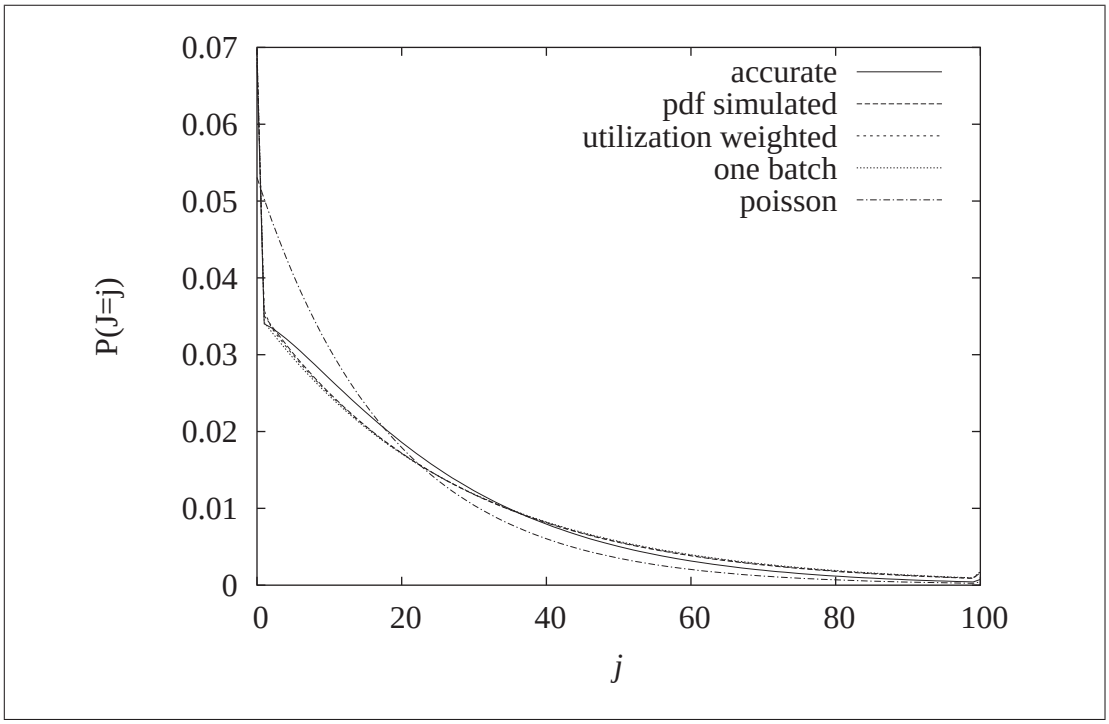


Figure 5.11: Equilibrium SOPs of the second queue for example network 1.

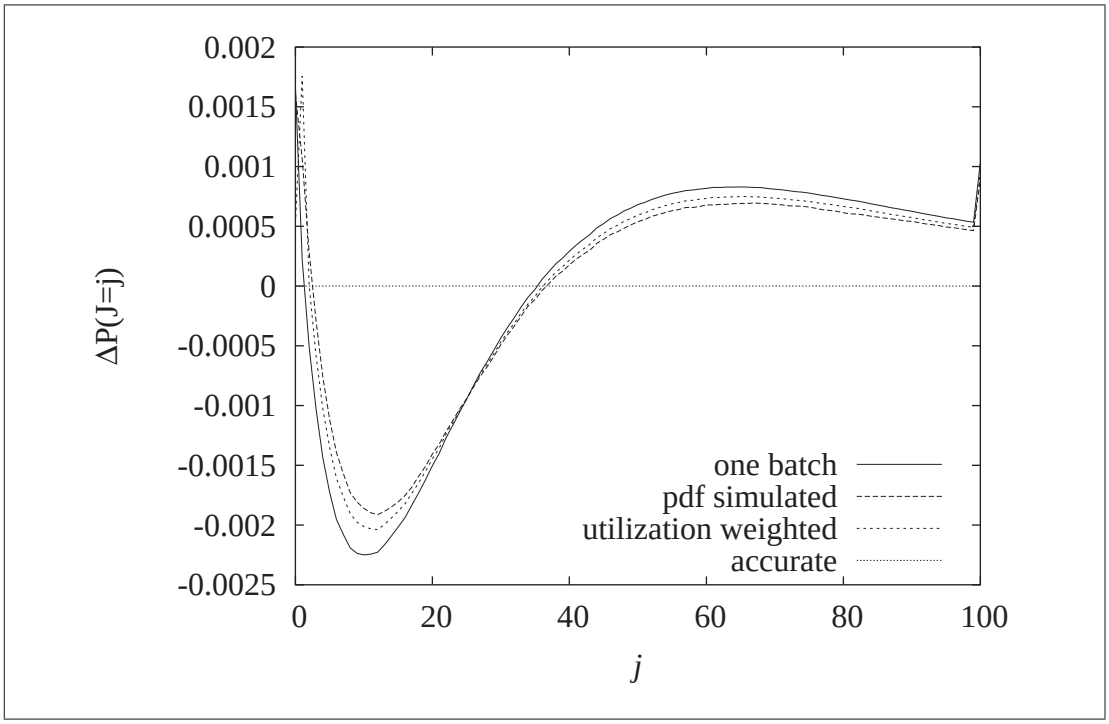


Figure 5.12: Errors of approximations methods in equilibrium SOPs of the second queue for example network 1.

were calculated via the detailed, job-based simulator. We see that in relative terms, the utilisation weighted traffic approximation is close to the pdf simulation, without any of the simulation overhead. The Poisson matching was omitted, as this was significantly worse than the other methods. Its inclusion would dramatically change the y-scale on the plots, losing much detail for the comparison of the more successful methods.

5.10 Loops

In a simple feed forward network, we achieve good results using the weighted approximation due to the absence of significant correlated traffic. Representing the departure traffic using exponential rates and geometric batch parameters does not introduce excessive errors in the queue length distributions, as the small difference between the pdf simulation and weighted matching results demonstrates. We now demonstrate the effects of correlated traffic on the accuracy of the solution by modifying the network. As illustrated in figure 5.13, we add a loop from the second queue back to the first, which is used by a departing customer with probability p. At the same time, we reduce the external arrival rate to some value λ, chosen so that the load at both queues is unchanged.

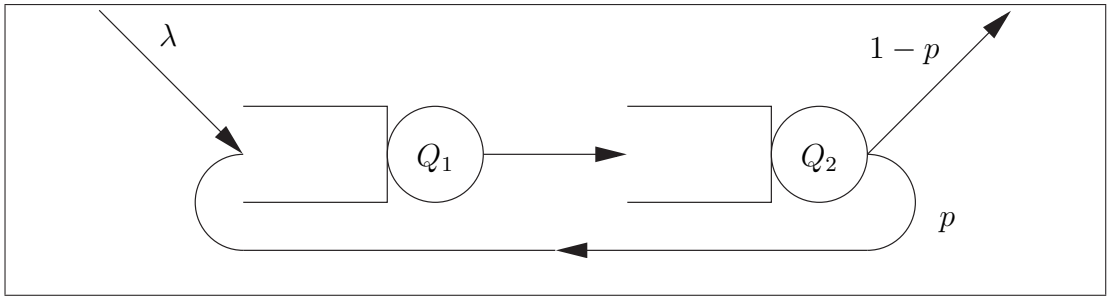


Figure 5.13: Topology of the example network with a loop.

For a loop probability of $p = 1/2$ with external arrival rate $\lambda = 9/10$ (giving total arrival rate to queue 1 of $9/5$), the relative error of the pdf-simulated solution is not significantly better than any of the iterative traffic matching methodologies (*c.f.* the figures 5.14). When the probability of looping is even higher, *e.g.* $p = 95/100$ (external arrival rate $\lambda = 9/100$, total arrival rate at queue 1 unchanged at $9/5$), the error

curves overlap. Here, in figures 5.15, the errors in the probability distributions are due to the strong correlation effects that the high loop probability introduces. Our inability to exactly describe departure traffic in terms of exponential distributions (and geometric distributions for batch sizes), is only a comparatively minor source of inaccuracy.

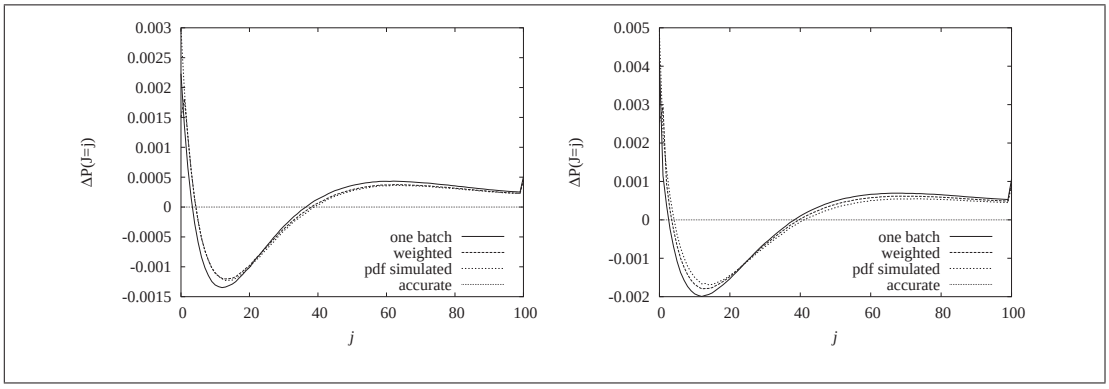


Figure 5.14: Errors of approximations methods in equilibrium SOPs of the first and second queue for the looping network with $p = 1/2$.

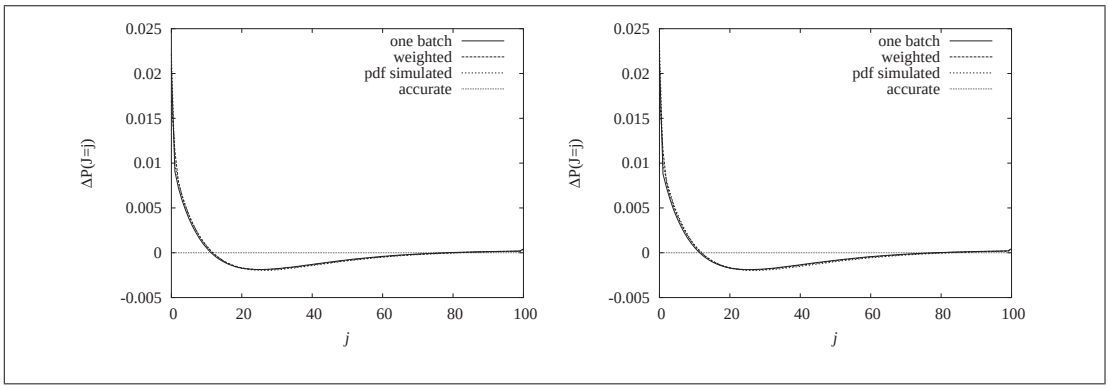


Figure 5.15: Errors of approximations methods in equilibrium SOPs of the first and second queue for the looping network with $p = 95/100$.

To achieve accurate results for our matching methods, we need to avoid networks with high loop probabilities and the associated correlated traffic, which we are not able to represent. Networks that do not satisfy this condition need to be solved using the accurate, job-based simulator.

5.11 Sojourn times on paths through networks

Chapter 4 gave sojourn time distributions for individual queues. When a customer passes from queue to queue along a path in a network, its time spent traversing the network is the sum of the times spent at each queue. If we attempt to derive distributions of the traversal time, the equivalent operation is to convolve the individual queue sojourn time distributions, assuming independence. The convolution operation is especially simple in the Laplace domain: a simple multiplication of the individual terms. We first re-use the simple feed-forward network in figure 5.10 to test this idea further in a network with little correlation between queues.

5.11.1 Network with Poisson link traffic

At first we assign unbatched rates to the external source and service processes. As a consequence the entire network exhibits a product form solution, *i.e.* it can be solved exactly for the steady-state by considering each queue in isolation. We can therefore expect the resulting network traversal time to be exact as well. The network is fed with external arrival rate $4/5$ and the infinite queues Q_1 and Q_2 have service rates of 1 and 2 respectively. Both queues are of type $M/M/1$ and their sojourn time distributions can be derived using the formulae in section 4.1.

For Q_1 we have $\lambda = 4/5$, $\mu = 1$ and $\kappa = 0$, hence its response time is

$$F_1(t) = 1 - e^{-\frac{t}{5}} \longleftrightarrow L_1(s) = \frac{1}{s} - \frac{1}{s + \frac{1}{5}}$$

There are no losses at Q_1 and its service process is Poisson, therefore for Q_2 we also have $\lambda = 4/5$. Further using $\mu = 2$ and $\kappa = 0$, we find

$$F_2(t) = 1 - e^{-\frac{6t}{5}} \longleftrightarrow L_2(s) = \frac{1}{s} - \frac{1}{s + \frac{6}{5}}$$

To find the probability density of the network sojourn time, we differentiate each term and form their convolution. Both of these operations are simplest using the Laplace transforms.

$$l_{net}(s) = (sL_1(s))(sL_2(s)) = \frac{\frac{1}{5} \times \frac{6}{5}}{(s + \frac{1}{5})(s + \frac{6}{5})} = \frac{6}{25}\left(\frac{1}{s + \frac{1}{5}} - \frac{1}{s + \frac{6}{5}}\right)$$

Inverting $l_{net}(s)$ gives $f_{net}(t) = \frac{6}{25}\left(e^{-\frac{t}{5}} - e^{-\frac{6t}{5}}\right)$. This analytical result is displayed alongside the simulated distribution in figure 5.16. As expected, for this entirely Pois-

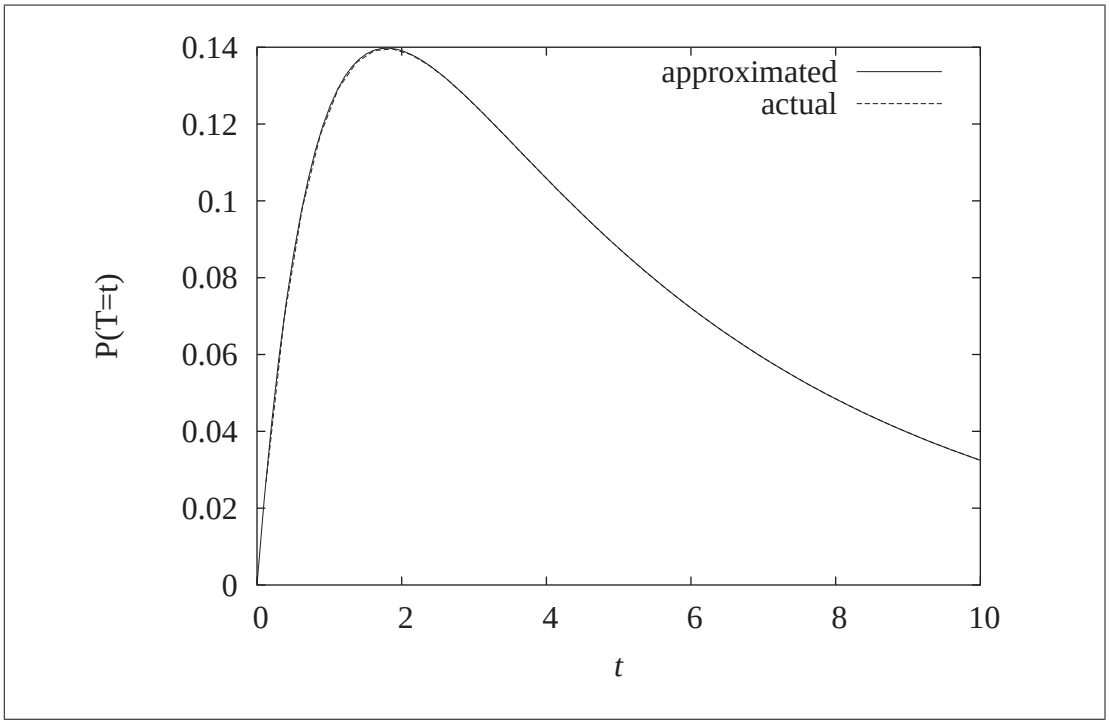


Figure 5.16: Actual and approximated network traversal distributions for a Poisson feed-forward network. The approximated result matches the simulation exactly.

son network, the approximated sojourn time is an exact match to the actual sojourn time.

5.11.2 Batched feed-forward network

We now modify the network so that a product form does not exist and exact state occupation probabilities are not available. The external arrival traffic is of rate 1 with geometric batch parameter $1/2$. Both queues have an infinite waiting room and share the same service characteristics: service rate 2, with batch parameter $1/4$. Figure 5.17 shows how both the one batch and weighted approximations are a good fit to the actual probability distributions. Somewhat surprisingly, the approximation that delivered a worse fit to the steady-state state occupation probability results in a slightly superior sojourn time distribution for this example.

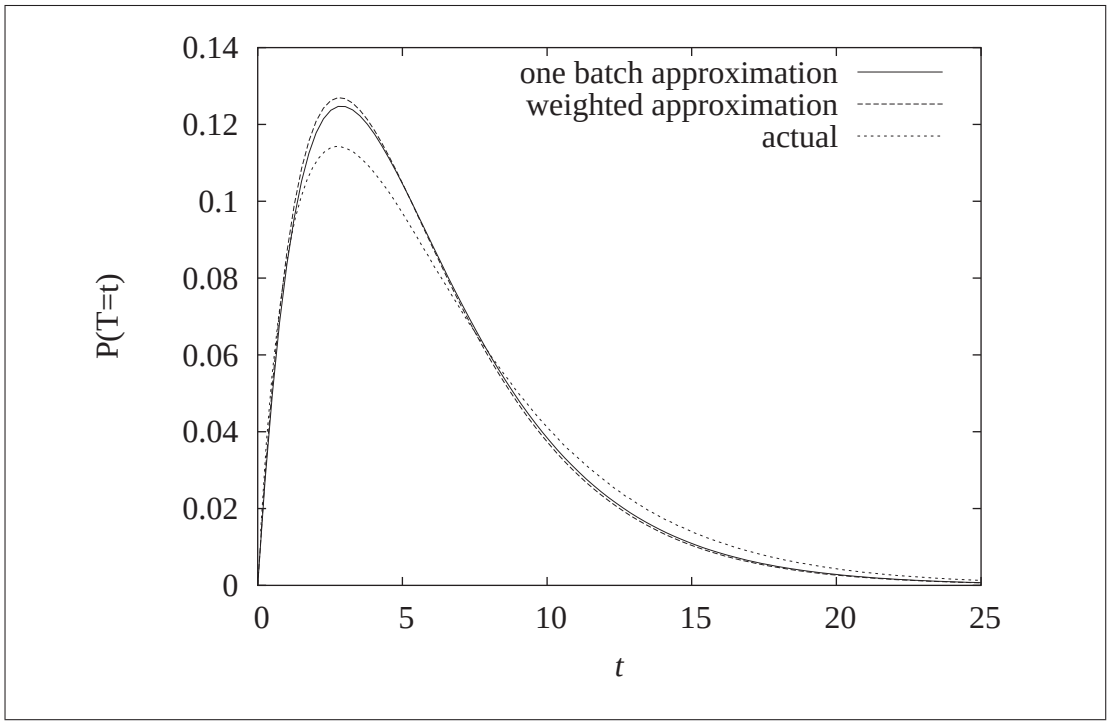


Figure 5.17: Actual and approximated network traversal distributions for a feed-forward network. Good accuracy is achieved by the traffic matching methods.

5.11.3 Network with significant correlation

To show the severe inaccuracies that arise in the sojourn time distributions when only low-grade approximations to the equilibrium distributions are available, we once again peruse the looping network topology in figure 5.13. From the steady-state analysis it is already known that such a system exhibits a large measure of correlation which is not conducive to accurate equilibrium solutions. We keep the service parameters from the feed forward example and adjust the external arrival rate to $1/20$ so that the overall load on the two queues remains unchanged in relation to the previous feed-forward network.

Let $L_1(s)$ be the Laplace transform of the sojourn time distribution for Q_1, and $L_2(s)$ similarly for Q_2. The sojourn time distribution for a customer, given that it does not loop is then given as $L_{both}(s) = L_1(s) \times L_2(s)$. When the loop is used exactly once we convolve $L_1(s)$ and $L_2(s)$ again, giving the expression

$$L_1(s) \times L_2(s) \times L_1(s) \times L_2(s) = L_{both}(s) \times L_{both}(s)$$

More generally, the sojourn distribution for completing exactly i loops is $L_{both}^{i+1}(s)$. We now weight these on the probabilities for completing the loop i times (remember that the loop probability is $p = 9/10$) and obtain

$$L(s) = \sum_{i=0}^{\infty} (1-p)p^i L_{both}^{i+1}(s) = (1-p)\frac{L_{both}(s)}{1 - pL_{both}(s)}$$

The individual $L_{both}(s)$ are rational functions in s and hence the derived function $L(s)$ is also of rational form. Again, partial fractions allow us to convert $L(s)$ into a form that is easily invertible to yield $f(t)$, the approximated unconditional network traversal probability density. In figures 5.18 and 5.19 this quantity is shown alongside the actual, simulated distribution. It is clear that the peak probabilities and especially the tail of the distribution exhibit poor accuracy. The errors due to correlated traffic have propagated from the equilibrium state occupancy solution to the sojourn time calculation. In these situations, it is therefore necessary to use simulated results for the state occupation probabilities. The simulator already provides sojourn time results, so that at this point, there is no additional benefit of carrying out symbolic analysis on the sojourn time distributions in an environment with high correlation.

When simulation is too expensive, it might be possible to use a tool such as the Imperial PEPA compiler [BraDin+03a]. PEPA representations of entire networks can cause the problem of state space explosion. Infinite queues in particular are impossible to analyse using this tool, yet by choosing an appropriately large, but finite waiting rooms sizes, their behaviour can be approximated. Due to the aforementioned state space explosion, even parallel implementations like HYDRA [DinKno03, BraDin+03b] can quickly become more time-intensive than using the simulator presented in section 5.7.

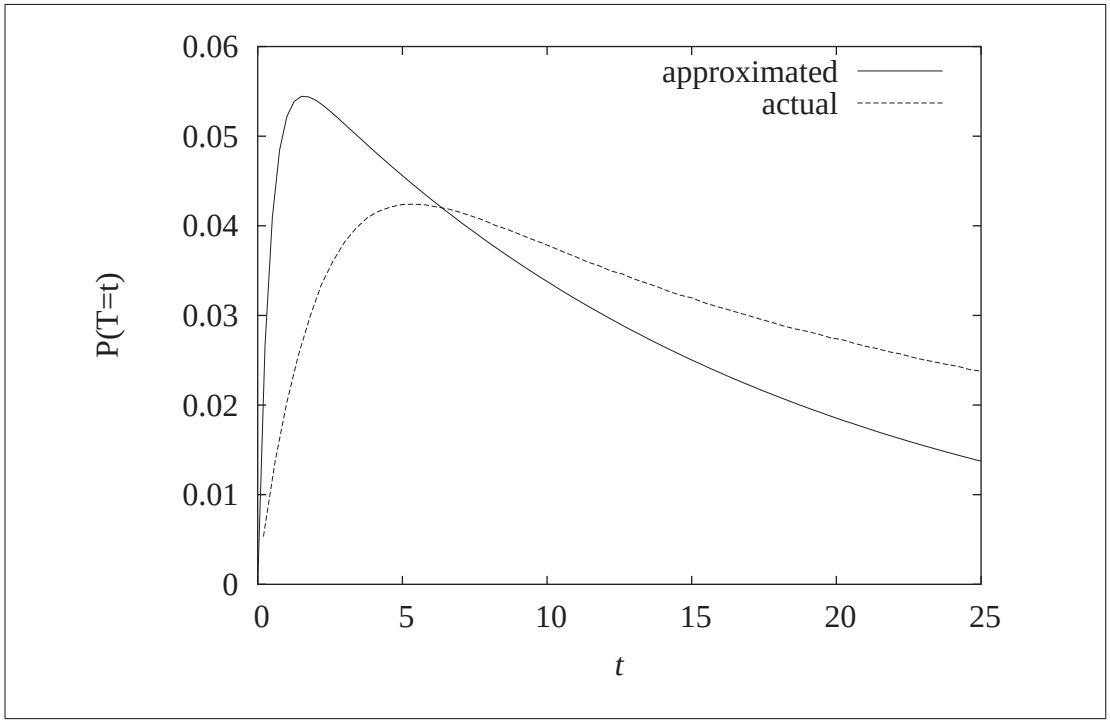


Figure 5.18: Actual and approximated network traversal distributions for a looping network. Region around the peak probabilities.

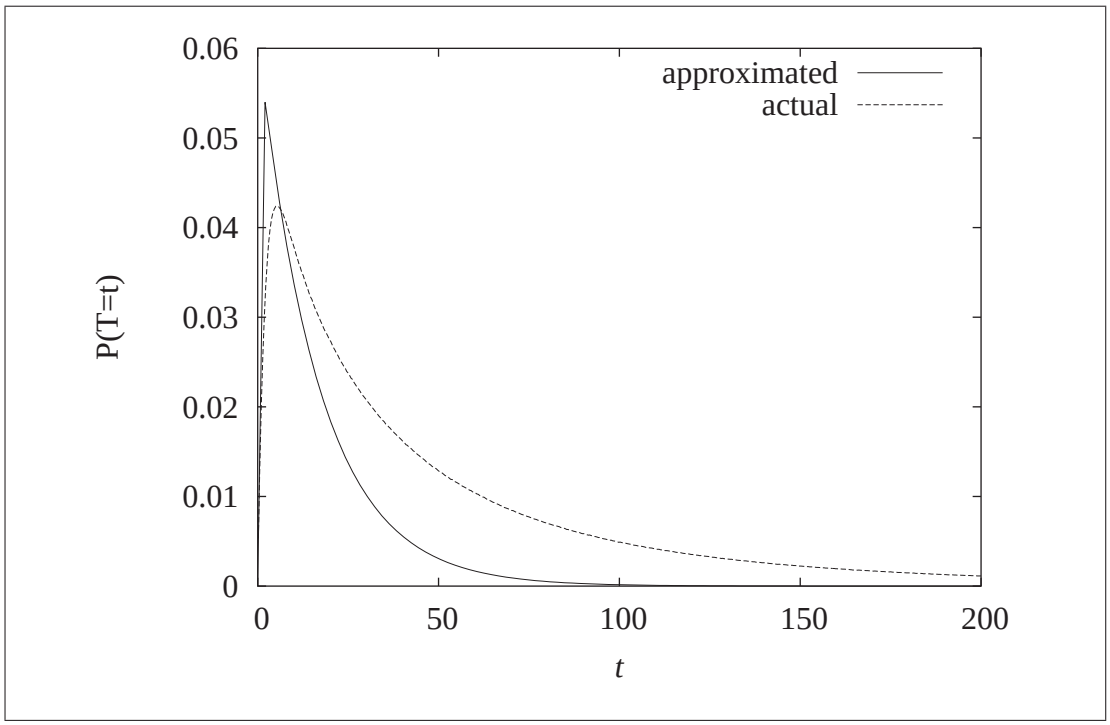


Figure 5.19: Actual and approximated network traversal distributions for a looping network. Overview of the distributions.

Chapter 6

Conclusion

6.1 Summary of Thesis Achievements

In this thesis we have addressed the challenge of providing accurate performance metrics for processing units modelled by a descriptive class of queues in an efficient manner. The particular features of the queue model, in particular geometrically batches with distributed size, modulation and multiple streams, allow for a more faithful reproduction of real-world systems than previously.

Steady-state solution methods for these queues require modified, localised balance equations to be computationally effective. Added complexity in the queue model makes the derivation of these balance equations by hand an error-prone process. In cooperation with David Thornley, we have developed an automated methodology that, given the system parameters, provides the user with localised balance equations and their solution. This new algorithm that we proposed is detailed in Appendix A and for the first time allows a rapid and error-free environment in which such queues can be studied.

Conventional methods for the derivation of steady-state state occupation probabilities, such as the classic matrix geometric (MG) method and spectral expansion (SE), fail in some cases because of the added complexity in the model. The generalised spectral expansion method in chapter 3 was the second major contribution of this work. We

have proved that it achieves the correct equilibrium solutions in all cases, including those where MG and SE would not be successful. As a by-product, the relative utilities of the MG and SE methods was assessed quantitatively. In addition we have described a modification that allows the matrix geometric method to be used as well, so that its many associated algorithms are still available. A Hybrid method is proposed that uses MG for efficiency and then switches to GSE to improve expressiveness and accuracy of the solution.

The sojourn time distributions for RCH queues derived in chapter 4 are the third major contribution of this work. Analytical models have struggled to provide precise estimates in the time domain and Laplace transforms of density functions have often been the best that can be determined. Although useful for calculating moments of distributions and often numerically invertible, quantiles can usually only be found reliably in the mid-range; in the crucial tail of a density function, numerical inversion quickly becomes unstable. By showing that the relevant Laplace transforms take rational, analytically invertible form we have obtained the full time-domain distribution functions.

Chapter 5 displayed a range of approximate methods for the steady-state analysis of entire networks of geometrically batched queues with Markov modulation. Encouraging accuracy is achieved for those networks that do not exhibit a large amount of correlated traffic caused by loops in the network. Solution validation was provided by two types of network simulators, which clearly showed that errors caused by correlation cannot be countered by a more faithful representation of inter-queue traffic. The utilisation weighted traffic matching methodology we presented requires significantly fewer computational resources, yet is only marginally inferior to the pdf-based simulator.

6.2 Applications

In this section we review potential practical applications of the theory and algorithms developed in this thesis.

The underlying queue structure for which we have developed both steady-state and

response time distributions, is that of a Markov modulated, multi-server queue with arbitrarily many arrival, service and killing streams, all of which can be geometrically batched. These are all desirable features when modelling *e.g.* multi-service traffic on the Internet. For other communication media such as mobile networks, the three main data types voice, text and images, can be represented by different amounts of burstiness and superimposed on the same channel to simulate the coexistence within shared bandwidth.

We have developed an automated solution framework for a class of queues, similar to the one used in an ad hoc way by Chakka et al in [ChaDo+03], who give further application areas. They model the effects of Ethernet link aggregation, where a server is connected to a switch through multiple, not necessarily similar network interface cards. A slightly modified queue with heterogeneous servers is also shown to model so-called optical burst switching nodes. Here the quantity c, that is otherwise used for the number of servers, gives the number of wavelengths and $L - c$ is the buffer size for the optical switch.

Traditionally there has been effort on the side of performance analysts to derive an accurate modelling description of real-world networks, yet all models have a limit beyond which an accurate solution is not possible anymore — such as when heavy correlation is encountered with the methods in this thesis. In industry [Dav+02] a device has been developed that is inserted into physical networks, which can shape traffic by delaying certain packets. When used to convert traffic to a nature similar to Poisson streams, many analytical performance measures, such as those in this thesis become accurate. The trade-off for this presence of more accurate analytical models is of course that the delays that are introduced into the actual, physical model.

6.3 Future Work

This section discusses ideas and suggestions for work that could be undertaken to take the content of this thesis further.

Our weighted traffic approximation represents traffic stream using two superimposed

Poisson streams, one of which is also batched. The accuracy achieved is impressive considering this relative simplicity. By designing a more complex traffic model, using more streams that are additionally modulated, the accuracy gap between the pdf-simulated solution and the matching methods could be narrowed further.

Using the proposed modifications to the matrix geometric method we can achieve good accuracy for infinite queues, yet the sojourn time solutions we give require the solution in terms of eigenvalues and eigenvectors, so that this method is not directly applicable. An alternative formulation for the Laplace transform $L(s)$, that expresses the generating function $\underline{\mathbf{D}}(s,z)$ in terms of the MG-matrix $\underline{\mathbf{F}}$, would allow us to combine the efficiency of the iterative matrix geometric methods with the conciseness of sojourn time calculations.

We have given expressions for modulated traffic approximations, but not investigated how accurate a solution the iterative network solver provides in this case. The multitude of network topologies together with modulation structures in all traffic requires a vast number of cases to be considered. A comprehensive overview could allow more problems to be solved.

Response time distributions are only available for RCH-type negative customers. When negative customers remove jobs from the *tail* of the queue the Laplace transform expression are more complex and have not yet been successfully inverted. Derivation of distributions for this class of queues would complete the analysis of the type of queues we use.

Appendix A

Balance Equation Localisation

In chapter 2, we defined a function $\overrightarrow{\mathbf{r}}_j$ which, for a given level j, provides the (symbolic) left-hand side of a Kolmogorov (balance) equation $\overrightarrow{\mathbf{r}}_j = \overrightarrow{\mathbf{0}}$. These balance equations generally include probability vectors from all levels of the queue due to the presence of batch arrival and departure streams. We call these upward and downward streams respectively, referring to the direction of probability flux in the Kolmogorov balance equations. In order to efficiently solve for the steady-state solution of large finite or infinite systems, we *localise* balance equations using a process we call *stream elimination*. Its aim shares a goal with Gaussian elimination: that of "block-diagonalising" the system. A key difference is that stream elimination exploits the structure of the queue to enable it to solve infinite systems.

We note that the matrix Kolmogorov balance equations for individual levels can be taken as columns in a larger Matrix problem, as in expression A.1. In this expression, $\underline{\mathbf{\Omega}}_{j,k}$ is the matrix coefficient of $\overrightarrow{\mathbf{v}}_k$ in the balance equation for level j (*c.f.* chapter 2). With unbounded batch transitions, such as found with geometrically batched processes, all $\underline{\mathbf{\Omega}}_{j,k}$ are non-zero. Our automated system implicitly performs column operations on this system, using coefficients calculated from our characterisation of the relationships between off diagonal block terms in the system as specified before manipulation.

$$(\overrightarrow{\mathbf{v}}_0\ \overrightarrow{\mathbf{v}}_1\ \ldots\ \overrightarrow{\mathbf{v}}_L)\begin{pmatrix} \underline{\mathbf{\Omega}}_{0,0} & \underline{\mathbf{\Omega}}_{1,0} & \cdots & \\ \underline{\mathbf{\Omega}}_{0,1} & & & \\ \vdots & \ddots & & \\ & & & \underline{\mathbf{\Omega}}_{L,L} \end{pmatrix} = \overrightarrow{\mathbf{0}} \tag{A.1}$$

Note that this expression establishes the relationship between the state occupation probabilities to within a common multiple. We add a normalisation equation, summing the probabilities and equating to 1 to complete the solution. Once we have written down this complete set of equations, we can abstract away from the queueing problem itself, and manipulate the set of expressions directly. We do, however, make use of observations on the formulation of the problem to enable us to calculate the coefficients we use for the standard column operations on the matrix in (A.1). At least one of the pair of coefficient matrices used for the elimination is non-singular, so the system is not rendered singular by these operations.

If the system had included only unbatched streams, the matrix would be tri-diagonal. The wider off-diagonal terms are created by the geometric batching. Our goal is to eliminate these terms to create a system which we can represent more efficiently. This is achieved by reducing the system to a number of adjacent diagonals. For a finite system ($L \neq \infty$), this can be considered to map directly to a Gaussian elimination type of solution. When the system is infinite, we clearly cannot write down the bottom-right corner of the matrix in the expression, but we can represent it as a limit.

Each stream's set of geometric batching terms is eliminated by combining scalar multiples of two neighbouring left-hand side terms, which map to columns in the above expression. The *target* level (usually called j) is the level to which the resulting localised left-hand side pertains. The *eliminator* level provides the terms which will be subtracted to remove the stream.

A.1 Stream elimination example

We can eliminate any stream from a target level using any eliminator which has contributions of that same stream. For example, we can remove the summation terms associated with the stream of positive arrivals from any level using any other except level L (for a finite queue).

The closer the eliminator is to the target, the fewer terms for that stream remain. So, the choice is between using the level below, or the level above, and either would work for any given target level. There is in fact no choice, certainly for downward streams (and also for upward streams in a finite queue) if we consider the situation at the limits of the queue $j = 0$ and $j = L$, as we explain below.

The simplest process to consider is the arrival of batches of positive customers. These arrive at every level in the queue. The stream arriving at level L must be cancelled out using the level below, as there are none above. To cancel the positive arrival stream for target level $L-1$, we could use either the level above or below. To see that we cannot use the level above to get an independent equation, we will show that the determinant of a matrix representation of the solution would have a zero determinant. The balance equations for our class of queues can be written as follows:

$$\overrightarrow{\mathbf{v}}\,\underline{\mathbf{A}} = \overrightarrow{\mathbf{0}}$$

Where $\overrightarrow{\mathbf{v}}$ is the row vector $(\overrightarrow{\mathbf{v}}_1, \overrightarrow{\mathbf{v}}_2, \ldots, \overrightarrow{\mathbf{v}}_L)$, and the columns $(j-1)N+1$ through jN of $\underline{\mathbf{A}}$ give the matrix coefficients of the vector $\overrightarrow{\mathbf{v}}_j$ for the target level j. The matrix $\underline{\mathbf{A}}$ is as follows for an MM CPP/GE/1/4 queue with N modulation states:

$$\underline{\mathbf{A}} = \begin{pmatrix} (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}^2 & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}^3 \\ \underline{\mathbf{C}} & (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}^2 \\ \underline{\mathbf{C}}\underline{\mathbf{\Phi}} & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}}) & (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} \\ \underline{\mathbf{C}}\underline{\mathbf{\Phi}}^2 & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}} & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}}) & (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) \\ \underline{\mathbf{C}}\underline{\mathbf{\Phi}}^3 & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}}^2 & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}} & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}}) & (\underline{\mathbf{Q}}-\underline{\mathbf{C}}) \end{pmatrix}$$

Our transformation of the system involves column operations, which we can achieve by post-multiplication of $\underline{\mathbf{A}}$ by an appropriate matrix, say $\underline{\mathbf{B}}$. The rank of $\underline{\mathbf{A}}$ is $5N-1$, *i.e.* one less than the number of states. When a column is removed and replaced by an expression reflecting the normalisation constraint, the system is correctly specified, so

we can solve it. If we post-multiply by a singular matrix, the rank of the system will be reduced, and will no longer be soluble with the normalisation constraint. Thus, the determinant of $\underline{\mathbf{B}}$ must be non-zero. For example, to remove the arrival stream terms in column five, we could post-multiply by a matrix $\underline{\mathbf{B}}$:

$$\underline{\mathbf{B}} = \begin{pmatrix} \underline{\mathbf{I}} & 0 & 0 & 0 & 0 \\ 0 & \underline{\mathbf{I}} & 0 & 0 & 0 \\ 0 & 0 & \underline{\mathbf{I}} & 0 & 0 \\ 0 & 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} \\ 0 & 0 & 0 & 0 & \underline{\mathbf{I}} \end{pmatrix}$$

The determinant of $\underline{\mathbf{B}}$ is 1, so our solution is safe. This yields a modified $\underline{\mathbf{A}}$ matrix:

$$\begin{pmatrix} (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}^2 & 0 \\ \underline{\mathbf{C}} & (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} & 0 \\ \underline{\mathbf{C}}\underline{\mathbf{\Phi}} & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}}) & (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}}) & 0 \\ \underline{\mathbf{C}}\underline{\mathbf{\Phi}}^2 & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}} & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}}) & (\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}}) & \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}}-\underline{\mathbf{\Theta}})-(\underline{\mathbf{Q}}-\underline{\mathbf{\Lambda}}-\underline{\mathbf{C}})\underline{\mathbf{\Theta}} \\ \underline{\mathbf{C}}\underline{\mathbf{\Phi}}^3 & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}}^2 & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}} & \underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}}) & (\underline{\mathbf{Q}}-\underline{\mathbf{C}})-\underline{\mathbf{C}}(\underline{\mathbf{I}}-\underline{\mathbf{\Phi}})\underline{\mathbf{\Theta}} \end{pmatrix}$$

Now consider the removal of the positive arrival components from column 4 (queue level 3) *in addition*. We can achieve this by post multiplying A by one of the two following B matrices:

$$\underline{\mathbf{B}}_{\text{bad}} = \begin{pmatrix} \underline{\mathbf{I}} & 0 & 0 & 0 & 0 \\ 0 & \underline{\mathbf{I}} & 0 & 0 & 0 \\ 0 & 0 & \underline{\mathbf{I}} & 0 & 0 \\ 0 & 0 & 0 & \underline{\mathbf{\Theta}} & -\underline{\mathbf{\Theta}} \\ 0 & 0 & 0 & -\underline{\mathbf{I}} & \underline{\mathbf{I}} \end{pmatrix}, \underline{\mathbf{B}}_{\text{ok}} = \begin{pmatrix} \underline{\mathbf{I}} & 0 & 0 & 0 & 0 \\ 0 & \underline{\mathbf{I}} & 0 & 0 & 0 \\ 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} & 0 \\ 0 & 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} \\ 0 & 0 & 0 & 0 & \underline{\mathbf{I}} \end{pmatrix}$$

These remove upward streams of positive arrivals from the balance equations for levels 3 and 4. $\underline{\mathbf{B}}_{\text{ok}}$ removes the arrivals using the level above, and $\underline{\mathbf{B}}_{\text{bad}}$ uses the level below as eliminator. Since columns four and five are linearly dependent, the determinant of $\underline{\mathbf{B}}_{\text{bad}}$ is 0, so use of this matrix in multiplication would render the equation ensemble degenerate. The determinant of $\underline{\mathbf{B}}_{\text{ok}}$ is 1, so it is safe to use.

A matrix $\underline{\mathbf{B}}_{\text{all}}$ to remove all upward streams could be either of the following (since removing the upward stream at level 1 is not necessary):

$$\underline{\mathbf{B}}_{\text{all}} = \begin{pmatrix} \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} & 0 & 0 & 0 \\ 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} & 0 & 0 \\ 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} & 0 \\ 0 & 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} \\ 0 & 0 & 0 & 0 & \underline{\mathbf{I}} \end{pmatrix} \text{ or } \begin{pmatrix} \underline{\mathbf{I}} & 0 & 0 & 0 & 0 \\ 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} & 0 & 0 \\ 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} & 0 \\ 0 & 0 & 0 & \underline{\mathbf{I}} & -\underline{\mathbf{\Theta}} \\ 0 & 0 & 0 & 0 & \underline{\mathbf{I}} \end{pmatrix}$$

Thus, we are constrained to eliminating upward streams using the balance equation at one level lower than the target level. A similar argument demands removal of downward streams using the raw left-hand side from the level above.

The expression resulting from elimination of a stream includes terms from all the queue levels of the constituent left-hand sides. In each modulation state m, there

are a number, say u_m, of distinct upward streams to cancel out, which requires the use of left-hand sides from levels $j - u_m \ldots j - 1$. Similarly in the general case[1], there are d_m distinct downward streams to this level, which requires eliminators from $j + 1 \ldots j + d_m$. The (vector) balance equations cover the maximum range over the modulation states upward and downward. We define $\hat{u}$ as the maximum over m of u_m, and $\hat{d}$ similarly. A localised equation for level j therefore comprises levels $j - \hat{u}$ through $j + \hat{d}$.

A.2 The recursive elimination operator

We define the transformation operator $E^{\nu,X}_{\tau,j}$, to be applied to a left-hand side vector-valued *function* $\overrightarrow{\mathbf{r}}$ (mapping integers j to left-hand sides $\overrightarrow{\mathbf{r}}_j$). This returns a left-hand side without batch sums of the stream given by the diagonal matrix of rates $\underline{\mathbf{X}}$, *e.g.* $\underline{\mathbf{X}} = \underline{\mathbf{\Lambda}}$:

$$E^{\nu,\underline{\mathbf{X}}}_{\tau,j}\overrightarrow{\mathbf{r}} = \overrightarrow{\mathbf{r}}_j \frac{\partial \overrightarrow{\mathbf{r}}_{j+\nu}}{\partial(\overrightarrow{\mathbf{v}}_\tau \underline{\mathbf{X}})} - \overrightarrow{\mathbf{r}}_{j+\nu} \frac{\partial \overrightarrow{\mathbf{r}}_j}{\partial(\overrightarrow{\mathbf{v}}_\tau \underline{\mathbf{X}})} = \overrightarrow{\mathbf{r}}_j \underline{\mathbf{A}} + \overrightarrow{\mathbf{r}}_{j+\nu} \underline{\mathbf{B}} \qquad \text{(A.2)}$$

The index j is the target level for which the returned function is a localised Kolmogorov equation. In it, batches of the stream with matrix rate term $\underline{\mathbf{X}}$ are removed by linear combination of the balance equation left-hand sides given by $\overrightarrow{\mathbf{r}}$ at j and at the elimination level $j + \nu$. The weightings are determined by the partial derivatives which serve only to select the appropriate coefficients. We therefore eliminate downward streams (processing completions and the action of negative customers) by using terms above the target, so $\nu = 1$ for these. Upward streams (positive customer arrivals) are eliminated using terms below the target so $\nu = -1$ for these.

The index τ is the queue level at which the coefficients of $\underline{\mathbf{X}}$ are calculated, and we call this the *test level*. This must be outside the range of levels whose left-hand sides are used in constructing the localised left-hand side for level j. We call this range the *elimination bracket*.

[1] within the processing region, tail-safe negative customers cannot operate, resulting in fewer downward streams within this region.

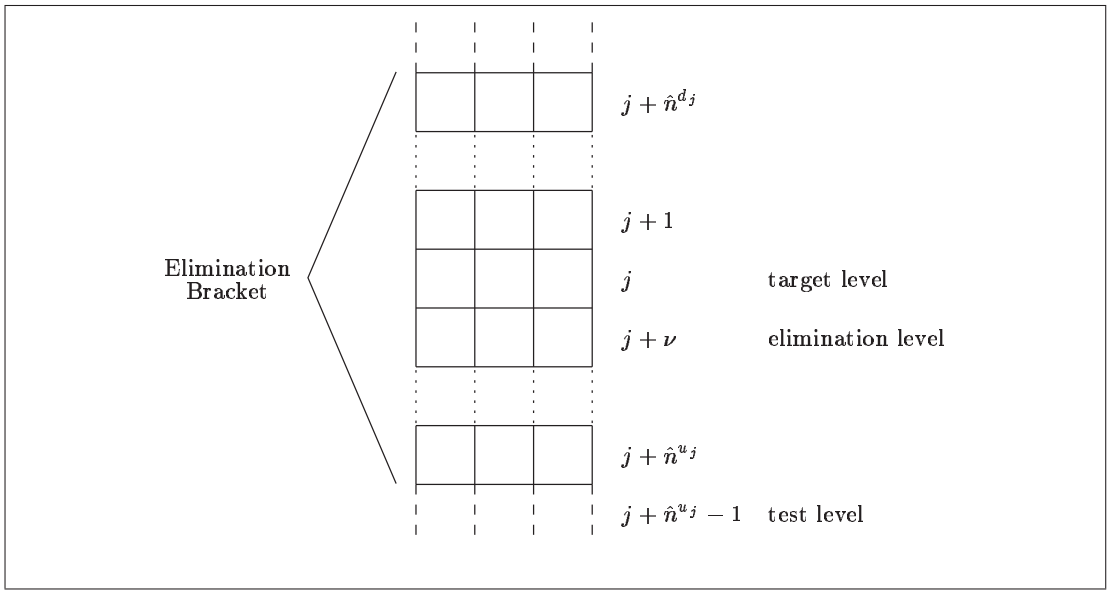


Figure A.1: Elimination of an upward stream ($\nu = -1$)

There may be a non-zero common factor between the two resulting elimination terms $\underline{\mathbf{A}}$ and $\underline{\mathbf{B}}$ in equation (A.2), and we can divide through by this because we ultimately equate the resulting left-hand side to zero. For example, eliminating the positive arrival stream in an MM CPP/MM/1 queue with only batched positive arrivals for level 2 involves the following terms:

$$\begin{aligned}
\frac{\partial \overrightarrow{\mathbf{r}}_2}{\partial(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}})} &= \frac{\partial}{\partial(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}})}\Big(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} + \overrightarrow{\mathbf{v}}_1\underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}) + \overrightarrow{\mathbf{v}}_2[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{C}}] + \overrightarrow{\mathbf{v}}_3\underline{\mathbf{C}}\Big) \\
&= (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} \\
\frac{\partial \overrightarrow{\mathbf{r}}_1}{\partial(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}})} &= \frac{\partial}{\partial(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}})}\Big(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}) + \overrightarrow{\mathbf{v}}_1[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{C}}] + \overrightarrow{\mathbf{v}}_2\underline{\mathbf{C}}\Big) \\
&= (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})
\end{aligned}$$

We use these in the elimination function as follows:

$$\begin{aligned}
E_{0,2}^{-1,\mathbf{\Lambda}}\overrightarrow{\mathbf{r}} &= \overrightarrow{\mathbf{r}}_2\frac{\partial \overrightarrow{\mathbf{r}}_1}{\partial(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}})} - \overrightarrow{\mathbf{r}}_1\frac{\partial \overrightarrow{\mathbf{r}}_j}{\partial(\overrightarrow{\mathbf{v}}_0\underline{\mathbf{\Lambda}})} \\
&= \overrightarrow{\mathbf{r}}_2(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}}) - \overrightarrow{\mathbf{r}}_1(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}} \\
&= \overrightarrow{\mathbf{r}}_2 - \overrightarrow{\mathbf{r}}_1\underline{\mathbf{\Theta}}, \text{ post-multiplying by the inverse of the common factor} \\
&= \overrightarrow{\mathbf{v}}_1[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] + \overrightarrow{\mathbf{v}}_2[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}})] + \overrightarrow{\mathbf{v}}_3\underline{\mathbf{C}}
\end{aligned}$$

A.3 Degenerate streams during elimination

Apart from reliable automation, a key advantage our methods offer is the accommodation of arbitrary numbers of independently distributed batched streams across modulation states. If an entirely pre-derived matrix balance equation is used (*e.g.* [ChaDo+03]), solution the system can become degenerate if two streams in a modulation state have identical batch parameters, and unstable if they are very similar. These circumstances arise naturally during the process of solving for the steady-state of a network. It is therefore essential that the queue solution mechanism can cope with such potential degeneracy. In this section, we explain our method's response to the circumstance of identical batching.

For each arrival or service completion process, it is possible to have a mixture of null (having zero rate, or a zero batch distribution parameter) and non-null batched streams across the modulating states. When performing the E-transformation, for a phase with zero rate or batch parameter, the corresponding component of the vector left-hand side returned is simply the argument's component at j. This is because there are no non-local terms to remove. For example, to eliminate a negative customer stream from the raw balance left-hand side for a finite t^v queue, if $\delta_1 = 1/2$ and $\delta_2 = 0$, we have (taking $\overrightarrow{\mathbf{r}}_{j,m}$ to be the m^{th} element of the vector returned by $\overrightarrow{\mathbf{r}}_j$):

$$E_{L,j}^{+1,R}\overrightarrow{\mathbf{r}} = \left(\underbrace{\overrightarrow{\mathbf{r}}_{j,1}\frac{\partial\overrightarrow{\mathbf{r}}_{j+1,1}}{\partial(\overrightarrow{\mathbf{v}}_L\delta_1)} - \overrightarrow{\mathbf{r}}_{j+1,1}\frac{\partial\overrightarrow{\mathbf{r}}_{j,1}}{\partial(\overrightarrow{\mathbf{v}}_L\delta_1)}}_{\substack{\text{elimination was required}\\ \text{in this modulation state}}}, \underbrace{\overrightarrow{\mathbf{r}}_{j,2}}_{\text{no elimination}} \right)$$

Examining the elements of matrices A and B in equation A.2, we note that whenever it is not necessary to eliminate a stream in modulation state m, the m^{th} elements of the diagonals of A and B are informative, as follows. Element $a_{m,m}$ is zero if the stream has already been removed. The value of $b_{m,m}$ is zero if either the stream has been removed, or the batch parameter of that stream is zero. The only case which requires care is when the stream has already been removed, wherein both $a_{m,m}$ and $b_{m,m}$ are zero, and we simply set $a_{m,m}$ to 1 if it is calculated in equation A.2 as zero. This keeps the term from the source equation at the target level, this being $\overrightarrow{\mathbf{r}}_{j,2}$ in the example

above.

Equation A.2 gives us a left-hand side *function of j* with stream $(\underline{\mathbf{X}}, \nu)$ removed. Elimination of further streams is simply the application of the same elimination procedure to the intermediate partially localised left-hand sides.

A.4 Worked example

We now perform a dry run of generating the localised balance equation for level 11 of a $c = 10$, t^v infinite waiting room MM CPP/GE/c/L G-queue. Equation A.2 is used recursively. This creates branches which encompass r_{10} through r_{13}. Thus, the test level for upward streams must be lower than 10 and the test level for downward streams must be higher than 13. We choose $\tau^u = 14$ and $\tau^d = 9$ as these are the values used in normal operation of the system.

In our elimination terminology, this localised balance equation is

$$\overrightarrow{\mathbf{b}}_{11} = E^{+1,\underline{\mathbf{K}}}_{14,11} E^{+1,\underline{\mathbf{C}}}_{14} E^{-1,\underline{\mathbf{\Lambda}}}_{10} \overrightarrow{\mathbf{r}}$$

The $\underline{\mathbf{K}}$ (negative customer arrival stream) elimination uses equations at levels 11 and 12, which in turn involve application of the $\underline{\mathbf{C}}$ (service completion) elimination term at, respectively, 11 and 12, and 12 and 13, and each of these involves an application of the $\underline{\mathbf{\Lambda}}$ (positive arrivals) elimination term, which in total uses the raw balance equations for levels 10 through 13.

First, we calculate a leaf of this tree, eliminating the positive customer arrival summation terms by taking the raw balance equations (as developed in section 2.7) for levels

10 and 11 and combining them appropriately.

$$\begin{aligned}
\overrightarrow{\mathbf{r}}_{10} = \ldots + \quad & \overrightarrow{\mathbf{v}}_9[\underline{\mathbf{\Lambda}}\overbrace{(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})}^{\frac{\partial \overrightarrow{\mathbf{r}}_{10}}{\partial(\overrightarrow{\mathbf{v}}_9\underline{\mathbf{\Lambda}})}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{10}[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{K}} - \underline{\mathbf{C}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}}) + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\mathbf{R} + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\mathbf{\Phi}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\mathbf{R}^2 + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\mathbf{\Phi}^2] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\mathbf{R}^2 + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\mathbf{\Phi}^3] + \ldots
\end{aligned}$$

$$\begin{aligned}
\overrightarrow{\mathbf{r}}_{11} = \ldots + \quad & \overrightarrow{\mathbf{v}}_9[\underline{\mathbf{\Lambda}}\overbrace{(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}}^{\frac{\partial \overrightarrow{\mathbf{r}}_{11}}{\partial(\overrightarrow{\mathbf{v}}_9\underline{\mathbf{\Lambda}})}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{10}[\underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{K}} - \underline{\mathbf{C}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}}) + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\mathbf{R} + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\mathbf{\Phi}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\mathbf{R} + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\mathbf{\Phi}^2] + \ldots
\end{aligned}$$

The coefficients of $\underline{\mathbf{\Lambda}}$ for level 9 are $\frac{\partial \overrightarrow{\mathbf{r}}_{10}}{\partial(\overrightarrow{\mathbf{v}}_9\underline{\mathbf{\Lambda}})} = (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})$ and $\frac{\partial \overrightarrow{\mathbf{r}}_{11}}{\partial(\overrightarrow{\mathbf{v}}_9\underline{\mathbf{\Lambda}})} = (\underline{\mathbf{I}} - \underline{\mathbf{\Theta}})\underline{\mathbf{\Theta}}$, hence in the ratio $\underline{\mathbf{I}} : \underline{\mathbf{\Theta}}$, so subtracting $\overrightarrow{\mathbf{r}}_{10}\underline{\mathbf{\Theta}}$ from $\overrightarrow{\mathbf{r}}_{11}$, we obtain $\overrightarrow{\mathbf{u}}_{11} = E_{9,11}^{-1,\underline{\mathbf{\Lambda}}}\overrightarrow{\mathbf{r}}$:

$$\begin{aligned}
\overrightarrow{\mathbf{u}}_{11} = \quad & \overrightarrow{\mathbf{v}}_{10}[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})) - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}))] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}}) + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\underline{\mathbf{R}}(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}}) + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\underline{\mathbf{R}}^2(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}}) + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}}^2(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})] \quad + \ldots
\end{aligned}$$

And we can write down the expressions for u_{12} and u_{13} directly from this, as the constituent behaviours are identical:

$$\begin{aligned}
\overrightarrow{\mathbf{u}}_{12} = \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})) - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}))] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}}) + \underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\underline{\mathbf{R}}(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}}) + \underbrace{\underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})}_{\frac{\partial \overrightarrow{\mathbf{u}}_{12}}{\partial(\overrightarrow{\mathbf{v}}_{14}\underline{\mathbf{C}})}}] \quad + \ldots \\
\overrightarrow{\mathbf{u}}_{13} = \quad & \overrightarrow{\mathbf{v}}_{12}[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{Q}} - \underline{\mathbf{\Lambda}} - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})) - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}}))] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}}) + \underbrace{\underline{\mathbf{C}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})}_{\frac{\partial \overrightarrow{\mathbf{u}}_{13}}{\partial(\overrightarrow{\mathbf{v}}_{14}\underline{\mathbf{C}})}}] \quad + \ldots
\end{aligned}$$

In all of these, only the level immediately below the target level appears in the localised expression, just as would have been the case without batches.

To eliminate the processing stream, with rate matrix $\underline{\mathbf{C}}$, we take pairs of these equations and provide them to the elimination operator $E^{+1,\underline{\mathbf{C}}}_{14,j}$ to be applied at levels $j = 11, 12, 13$. To calculate the expression for $E^{+1,\underline{\mathbf{C}}}_{14,11}\overrightarrow{\mathbf{u}}$ we find the coefficients of $\underline{\mathbf{C}}$ at the test level in $\overrightarrow{\mathbf{u}}_{12}$ and $\overrightarrow{\mathbf{u}}_{13}$, which are respectively $(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})\underline{\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})$ and $(\underline{\mathbf{I}} - \underline{\mathbf{\Phi}})(\underline{\mathbf{I}} - \underline{\mathbf{\Phi\Theta}})$. The ratio between these is $\underline{\mathbf{\Phi}} : \underline{\mathbf{I}}$, so we subtract $\overrightarrow{\mathbf{u}}_{12}\underline{\mathbf{\Phi}}$ from $\overrightarrow{\mathbf{u}}_{11}$ to yield the left-hand side $\overrightarrow{\mathbf{d}}$ with only $\underline{\mathbf{K}}$-batch terms:

$$\begin{aligned}
\overrightarrow{\mathbf{d}}_{11} = \quad & \overrightarrow{\mathbf{v}}_{10}[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta\Phi}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}) - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{R}} + \underline{\mathbf{\Phi}})) - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[-\underline{\mathbf{Q\Phi}} + \underline{\mathbf{\Lambda\Phi}} + \underline{\mathbf{K}}((\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R\Phi}}) + (\underline{\mathbf{I}} + \underline{\mathbf{\Theta}})\underline{\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})) + \underline{\mathbf{C}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}})(\underline{\mathbf{R}} - \underline{\mathbf{\Phi}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underbrace{\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})\underline{\mathbf{R}}(\underline{\mathbf{I}} - \underline{\mathbf{R\Theta}})(\underline{\mathbf{R}} - \underline{\mathbf{\Phi}})}_{\frac{\partial \overrightarrow{\mathbf{d}}_{11}}{\partial(\overrightarrow{\mathbf{v}}_{14}\underline{\mathbf{K}})}}] + \ldots \text{only terms in } \underline{\mathbf{K}}
\end{aligned}$$

Again, the behaviour for level 12 is identical, so we immediately have $\overrightarrow{\mathbf{d}}_{12}$:

$$\begin{aligned}
\overrightarrow{\mathbf{d}}_{12} = \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[\underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}\underline{\mathbf{\Phi}}) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}) - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} - \underline{\mathbf{R}} + \underline{\mathbf{\Phi}})) - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}})] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[-\underline{\mathbf{Q}}\underline{\mathbf{\Phi}} + \underline{\mathbf{\Lambda}}\underline{\mathbf{\Phi}} + \underline{\mathbf{K}}((\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R}}\underline{\mathbf{\Phi}}) + (\underline{\mathbf{I}} + \underline{\mathbf{\Theta}})\underline{\mathbf{\Phi}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})) + \underline{\mathbf{C}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{14}[\underbrace{\underline{\mathbf{K}}\,(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R}}\underline{\mathbf{\Theta}})(\underline{\mathbf{R}} - \underline{\mathbf{\Phi}})}_{\frac{\partial \overrightarrow{\mathbf{d}}_{12}}{\partial(\overrightarrow{\mathbf{v}}_{14}\underline{\mathbf{K}})}}] + \ldots \text{only terms in } \underline{\mathbf{K}}
\end{aligned}$$

In these expressions, only the level immediately above the target contains terms associated with processing completions, as would have been the case with unbatched processing.

Assuming unequal batch terms (*i.e.* $\underline{\mathbf{R}} \neq \underline{\mathbf{\Phi}}$), these can be combined using $E_{14,j}^{+1,\underline{\mathbf{K}}}\overrightarrow{\mathbf{d}}$ to eliminate the final batch term in $\underline{\mathbf{K}}$ by subtracting $\overrightarrow{\mathbf{d}}_{12}\underline{\mathbf{R}}$ from $\overrightarrow{\mathbf{d}}_{11}$ to give a localised balance $\overrightarrow{\mathbf{b}}_{11}$:

$$\begin{aligned}
\overrightarrow{\mathbf{b}}_{11} = \quad & \overrightarrow{\mathbf{v}}_{10}[\underline{\mathbf{\Lambda}} - (\underline{\mathbf{Q}} - \underline{\mathbf{K}} - \underline{\mathbf{C}})\underline{\mathbf{\Theta}}] \\
+ \quad & \overrightarrow{\mathbf{v}}_{11}[\underline{\mathbf{Q}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{R}})) - \underline{\mathbf{\Lambda}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}} + \underline{\mathbf{R}}) \\
& \qquad - \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}})) - \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}(\underline{\mathbf{I}} + \underline{\mathbf{R}}))] \\
+ \quad & \overrightarrow{\mathbf{v}}_{12}[-\underline{\mathbf{Q}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}\underline{\mathbf{\Phi}})) + \underline{\mathbf{\Lambda}}(\underline{\mathbf{\Phi}} + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}\underline{\mathbf{\Phi}})) \\
& \qquad + \underline{\mathbf{K}}(\underline{\mathbf{I}} + \underline{\mathbf{\Phi}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}})) + \underline{\mathbf{C}}(\underline{\mathbf{I}} + \underline{\mathbf{R}}(\underline{\mathbf{I}} + \underline{\mathbf{\Theta}}))] \\
+ \quad & \overrightarrow{\mathbf{v}}_{13}[\underline{\mathbf{Q}}\underline{\mathbf{\Phi}}\underline{\mathbf{R}} - \underline{\mathbf{\Lambda}}\underline{\mathbf{\Phi}}\underline{\mathbf{R}} - \underline{\mathbf{K}}\underline{\mathbf{\Phi}} - \underline{\mathbf{C}}\underline{\mathbf{R}}]
\end{aligned}$$

In this expression, we see two levels appear above the target level, one for each of negative customers and processing completions streams. This has the same form as the result found in [ChaHar01].

Note that the coefficient of $\overrightarrow{\mathbf{v}}_{13}$ is only strictly valid if it was necessary to remove all streams. The elimination ratio was determined from a term $\underline{\mathbf{K}}(\underline{\mathbf{I}} - \underline{\mathbf{R}})(\underline{\mathbf{I}} - \underline{\mathbf{R}}\underline{\mathbf{\Theta}})(\underline{\mathbf{R}} - \underline{\mathbf{\Phi}})$, which contains zeros wherever the diagonal elements of $\underline{\mathbf{R}}$ and $\underline{\mathbf{\Phi}}$ are equal. In these circumstances, the information required to formulate an elimination ratio is not present. This is an example of stream degeneracy treated in section A.3.

Appendix B

Brief semantics of PEPA

In chapter 5 we conducted our analysis using an abbreviated PEPA syntax, the full syntax and semantics of the PEPA language being given in [Hil94]. We have just four constructions:

1. The *prefix* combinator defines an agent $(a, \lambda).P$ that carries out action (a, λ) of *type* a at *rate* λ and subsequently behaves as agent P.
2. The agent describing the *cooperation* of two agents P and Q which synchronise over actions with types in a specified set L is written $P \underset{L}{\bowtie} Q$.
3. A new *constant* agent A is defined by the assignment combinator $A \stackrel{def}{=} P$ to have the same behaviour as P.
4. The *choice* combinator defines an agent $P + Q$ that can either be P or Q: which one is defined by a race condition between the currently enabled activity of P and Q.

Action hiding could easily be added to this syntax but is not used in this thesis.

In a cooperation $P \underset{L}{\bowtie} Q$, the agents P and Q proceed independently with any actions whose types do not occur in the cooperation set L. However, actions with types in L are only enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q. In standard PEPA, the shared action occurs at the rate of the slowest participant. Here, however, we assume that for each action type in a cooperation set L, exactly one agent (either P or Q) is *passive* and effectively its synchronising action has rate $\top = \infty$. This means

that the passive agent does not influence the rate at which a shared action occurs, essentially waiting for the other agent.

The value of a syntactic expression written in PEPA is given in terms of an underlying continuous time Markov chain semantic model, as in [Hil94] and summarised in [Har03]. The set of actions which an agent P may next engage in, the *current actions* of P, is denoted by $Ac(P)$, which can be defined inductively over the structure of P. When the system is behaving as agent P, these are the actions that are enabled. The new agents thus resulting from P are called the *derivatives* of P. If P can perform the action (a, λ) and then become P', we write $P \xrightarrow{(a,\lambda)} P'$ and say that P' is an a*-derivative* of P. The *derivative set*, denoted $ds(P)$, of an agent P is the transitive closure of all its derivatives and is defined by recursion. This defines a labelled transition system as a semantic model for PEPA.

The *derivation graph*, formed by syntactic PEPA terms (agents) at the nodes, with arcs representing the transitions between them, determines the underlying Markov process of an agent P. The *transition rate* between two agents C_i and C_j, denoted $q(C_i, C_j)$, is the sum of the action rates labelling arcs connecting node C_i to node C_j. In fact, a PEPA agent is associated not only with the continuous-time Markov chain defined by its derivation graph but also with one of the states in that graph, the *initial state*. If the derivation graph is irreducible, *i.e.* defines an irreducible Markov process, the choice of initial state is arbitrary at equilibrium, when this exists.

Glossary

blocking region Term for the queue levels at the top of a finite queue, that do not fall into the repeating region.

burstiness Quality of a stream that allows a batch of customers to perform the same transition. In this thesis, burstiness is modelled using geometric distributions.

eigenmode Term to denote an eigenvector with its associated eigenvalue.

equilibrium probability State occupation probability at steady-state.

exponential distribution Probability distribution that follows $P(T < t) = 1 - e^{-\lambda t}$. It is used to model delays between queue transitions and satisfies the memoryless property.

filling region Term for the queue levels at the bottom of a queue, that do not fall into the repeating region.

geometric distribution Probability distribution $P(s = S) = (1 - \theta)\theta^{s-1}$ that models the batch size of a stream.

G-Queue Term for a queue that experiences one or more killing streams.

h^p **head-per** The h^p killing paradigm selects a random processor of the queue and removes the customer that it is currently processing.

Kendall Notation that attempts to qualitatively describe a queue by using a series of fields, separated by $/$'s. Only the first three fields are commonly used and denote, in order, the arrival process, the departure process and the number of servers.

killing paradigm Method with which existing customers in a queue are removed by an incoming negative customer. The three modes in use are t^s tail safe, t^v tail vulnerable and h^p head-per.

killing stream Alternate term for a negative customer stream.

Kolmogorov balance equation These equations balance the total incoming rates to a state to the total outgoing rate from a state by equating them. That the Kolmogorov balance equations are satisfied is a necessary condition for a steady-state solution.

level (queue) Used to denote those states that have a common number of customers. For modulated queues this is a vector that encompasses all modulation states.

localised balance equation Transformation of a Kolmogorov balance equation that relates queue levels of a small, finite ranges as opposed to the possibly infinite ranges of the original Kolmogorov equations.

matrix geometric methods Set of methods for the steady-state solution of queues. It represents the changes in probability vectors between two adjacent levels using a matrix, *i.e.* $\overrightarrow{\mathbf{v}}_{j+1} = \overrightarrow{\mathbf{v}}_j \underline{\mathbf{F}}$.

Markovian When referring to a continuous quantity, denotes that a process follows an exponential distribution.

memoryless property *the future is independent of the past.* An exponentially distributed random variable satisfies $P(X > t + \delta_t | x > t) = P(x > \delta_t)$ for $t, \delta_t > 0$.

modulation (Markov) Markovian process that changes the behaviour of a queue it is associated with, by changing the transition rates. Can be used to model breakdowns and repairs as well as changes to an arrival process.

negative customer Customer, whose arrival at a queue causes existing customers to be removed (killed), with a consequential decrease in queue length. The manner in which the removal is performed depends on the killing paradigm in use.

repeating region Queue region within which the same matrix recurrence relation between neighbouring queue levels holds. It is bordered by the filling region below and, for a finite queue, by the blocking region above.

spectral expansion method Methods for the steady-state solution of queues. It represents the probability vectors explicitly using the set of eigenmodes of a matrix. *e.g.* $\overrightarrow{\mathbf{v}}_{j+i} = \sum_{k=1}^{n^*} \alpha_k \lambda_k^i \overrightarrow{\psi}_k$

steady-state Condition that a process reaches after a *sufficiently long time i.e.* after transient effect have disappeared. This is unachievable in practice, but is a very

good approximation to long-term behaviour.

stream Used as a group description for services as well as positive and negative arrivals, all of which cause transitions up or down in a queue.

t^s **tail safe** The t^s killing paradigm removes the last customer in the queue, unless it is already undergoing service.

t^v **tail vulnerable** The t^s killing paradigm removes the last customer in the queue, even if it is already in service.

Bibliography

[AsmNer+96] S. Asmussen, O. Nerman and M. Olsson. *Fitting Phase-Type Distributions vie the EM Algorithm.* Scandinavia Journal of Statistics, 23:419–441, 1996.

[BhaHar97] M. Bhabuta and P.G. Harrison. *Analysis of ATM traffic on the London MAN* Proceedings of the 4th International Conference on Performance Modelling and Evaluation of ATM Networks, Ilkely, Chapman and Hall, 1997.

[BinLat+02] D. Bini, G. Latouche and B. Meini *Solving matrix polynomial equations arising in queueing problems* in Linear Algebra and its Applications 340: 225-244 Jan 1 2002.

[BinMei97] D. Bini and B. Meini *Improved cyclic reduction for solving queueing problems* Numerical Algorithms, 15:57–74, 1997.

[BraDin+03a] J.T. Bradley, N.J. Dingle, S.T. Gilmore and W.J. Knottenbelt. *Derivation of passage-time densities in PEPA models using ipc: the Imperial PEPA Compiler* in Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03), pages 344-351, Orlando FL, USA, October 12th-15th 2003.

[BraDin+03b] J.T. Bradley, N.J. Dingle, S.T. Gilmore and W.J. Knottenbelt. *Extracting passage times from PEPA models with the HYDRA tool: A case study* In Proceedings of the 19th UK Performance Engineering Workshop (UKPEW'03), pages 79-90, Warwick, July 9th-10th 2003.

[Cha95] R. Chakka. *Performance and Reliability Modeling of Computer Systems Using Spectral Expansion* PhD thesis, University of Newcastle upon Tyne, 1995.

[ChaDo+03] R. Chakka, T.V. Do and Z. Pandi. *Generalised Markovian queues and application to performance analysis in telecommunications networks* In Proceedings, First International Working Conference On Performance Modelling And Evaluation Of Heterogeneous Networks (HET-NETs 2003).

[ChaHar01] R. Chakka and P.G. Harrison. *A Markov modulated multi-server queue with negative customers - The MM CPP/GE/c/L G-queue* Acta Informatica **37**(11-12), pp. 881-919, 2001.

[CoxIsh 80] D. Cox and V. Isham. *Point processes* Monographs on applied probability and statistics series. Chapman and Hall, London 1980. ISBN 0412219107.

[Dav+02] N. Davies et al. *Filtering data Flows* Patent EP1327333/WO0230061.

[Dei91] A.S. Deif. *Advanced Matrix Theory for Scientists and Engineers* (2nd ed.), Gordon and Breach Science Publishers, 1991.

[DenMar93] L. Deng and J.W. Mark. *Parameter Estimation for Markov Modulated Poisson Processes via the EM Algorithm with Time Discretization.* Telecommunication Systems, Vol. 1:321–338, 1993.

[DinKno03] N.J. Dingle, W.J. Knottenbelt and P.G. Harrison. *HYDRA:Hypergraph-based Distributed Response-time Analyser* in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03), pages 215-219, Las Vegas NV, USA, June 23rd-26th 2003.

[FisMei92] W. Fischer and K. Meier-Hellstern. *The Markov-Modulated Poisson Process (MMPP) Cookbook.* Performance Evaluation, 18:149–171, 1992

[Gel91] E. Gelenbe. *Product form queueing networks with negative and positive customers* in Journal of Applied Probability **28**, pp. 656-663, 1991.

[GeHar+03] H. Ge, U. Harder and P.G. Harrison. *Parameter estimation for MMPPs using the EM algorithm* in Proceedings, 19th UK Performance Engineering Workshop (UKPEW' 2003), University of Warwick.

[GilHil94] S. Gilmore and J. Hillston. *The PEPA workbench: A tool to support a process algebra-based approach to performance modelling* in Proceedings of the 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (G. Haring and G. Kotsis, eds.), vol 794 of Lecture Notes in Computer Science, pp. 353 368, Springer-Verlag, Vienna, May 1994.

[Har02] P.G. Harrison. *The MM CPP/GE/c/L G-queue: sojourn time distribution* in Queueing Systems: Theory and Applications, 2002.

[Har03] P.G. Harrison. *Turning back time in Markovian process algebra* in Theoretical Computer Science Vol. 290, pp 1947-1986, 2003.

[HarKno02] P.G. Harrison and W.J. Knottenbelt. *Passage Time Distributions in Large Markov Models* in Proceedings ACM SIGMETRICS 2002, Marina Del Rey, California, USA, June 2002, pp. 77-85.

[HarPat92] P.G. Harrison and N.M. Patel. *Performance Modelling of Communication Networks and Computer Architectures* Addison-Wesley, 1992. ISBN 0201544199

[HarTho+02] P.G. Harrison, D.J. Thornley and H. Zatschler. *Geometrically batched networks* in proceedings (ISCIS 17) Seventeenth International Symposium On Computer and Information Sciences October 28-30, 2002 University of Central Florida Orlando, Florida.

[HarZat04] P.G. Harrison and H. Zatschler. *Sojourn time distributions in modulated G-queues with batch processing* in Proceeding, 1st International Conference on Quantitative Evaluation of Systems (QEST) 2004, University of Twente.

[HavOst97] B.R. Haverkort and A. Ost. *Steady-state analysis of infinite stochastic petri nets: comparing the spectral expansion and the matrix-geometric method* in Proc. 7th International Workshop on Petri Nets and Perfor-

mance Models, pages 36-45, Saint Malo, France, 1997. IEEE Computer Society Press.

[Hil94] J. Hillston. *A Compositional Approach to Performance Modelling* PhD Thesis, University of Edinburgh, 1994.

[Jay57] E.T. Jaynes, *Information Theory and Statistical Mechanics* Physical Review 106, 1957 pp. 620-630

[Jor70] C. Jordan. *Traité des Substitutions et des Equations Algebriques* Paris, Gauthier-Villars, 1870.

[KleLin+02] A. Klemm, C. Lindemann, and M. Lohmann. *Traffic Modeling of IP Networks Using the Batch Markovian Arrival Process* 12th Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (Tools 2002), London, UK, Lecture Notes in Computer Science, Vol. 2324, 92-110, Springer 2002.

[KouAlm88] D. Kouvatsos and John Almond. *Maximum Entropy Two-Station Cyclic Queues with Multiple General Servers* Acta Informatica, Vol 25. pp241-267,Errata: 501, 787

[KouAwa03] D. Kouvatsos and I. Awan. *Entropy maximisation and open queueing networks with priorities and blocking* in Performance Evaluation 51 (2003) 191-227.

[Kre88] E. Kreyszig. *Advanced Engineering Mathematics* 6th Ed. Wiley 1988 p1035.

[LatRam93] G. Latouche and V. Ramaswami. *A logarithmic reduction algorithm for quasi-birth-death processes* in Journal of Applied Probability, 30: pp 650-674, 1993.

[LelTaq+94] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. *On the self-similar nature of Ethernet traffic (extended version).* IEEE/ACM Transactions on Networking, Vol. 2 No. 1:1–15, 1994.

[Mei87] K.S. Meier-Hellstern. *A Fitting Algorithm for Markov-Modulated Poisson Processes having two arrival rates.* European Journal of Operational Research, 29:370–377, 1987.

[Mit87] I. Mitrani. *Modelling of computer and communication systems* Cambridge Computer Science Texts 24, Cambridge University Press, 1987.

[MitCha95] I. Mitrani and R. Chakka. *Spectral expansion solution for a class of Markov models: Application and comparison with the matrix-geometric method* in Performance Evaluation **23** pp. 241-260, 1995.

[NaoKri+97] Naoumov, Krieger and Wagner. *Analysis of a multi-server delay-loss system with a general Markovian arrival process* Matrix-analytic methods in stochastic models, 1997.

[Neu71] M.F. Neuts. *A Queue Subject to Extraneous Phase Changes* in Advanced Applied Probability **3**, 1971.

[Neu81] M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models* Johns Hopkins University Press, Baltimore, 1981.

[PaxFlo95] V. Paxman and S. Floyd. *Wide-Area Traffic: The Failure of Poisson Modelling* IEEE/ACM Transactions on Networking, 1995.

[Pea65] M.C. Pease. *Mathematics in Science and Engineering* Volume 16. Academic Press Inc. 1965.

[PreTeu+92] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* 2nd ed., Cambridge Univ. Press, New York, 1992.

[RisSmi02] A. Riska, E. Smirni *MAMSolver: A matrix-analytic methods tool* in Proceeding of the 12th International Conference on Modeling Techniques and Tools, LNCS 2324, pages 205-211, Springer-Verlag, London, April 2002.

[Ryd96] T. Rydé n. *An EM Algorithm for Estimation in Markov-Modulated Poisson Processes.* Computational Statistics and Data Analysis, 21:431–447, 1996.

[Smi+76] B.T. Smith et al. *Matrix Eigensystem Routines – EISPACK Guide* 2nd ed., vol. 6 of Lecture Notes in Computer Science (New York: Springer-Verlag) 1976.

[ThoZat+03] D.J. Thornley, H. Zatschler and P.G Harrison. *An automated formulation of queues with multiple geometric batch processes* in Proceedings, First International Working Conference On Performance Modelling And Evaluation Of Heterogeneous Networks (HET-NETs 2003).

[ThoZat03] D.J. Thornley and H. Zatschler. *Analysis and enhancement of network solutions using geometrically batched traffic* in Proceedings, 19th UK Performance Engineering Workshop (UKPEW' 2003), University of Warwick.

[WilRei71] J.H. Wilkinson and C. Reinsch. *Linear Algebra* vol. II of Handbook for Automatic Computation (New York: Springer-Verlag) 1971.

[Wol99] S. Wolfram, *The Mathematica Book* 4th ed., (Wolfram Media/Cambridge University Press, 1999).