

# Analysis of Queueing Systems with Sample Paths and Simulation Companion

Nicky D. van Foreest

June 28, 2020



## CONTENTS

---



## INTRODUCTION

---

### 0.1 INTRODUCTION

#### *Motivation and Examples*

Queueing systems abound, and the analysis and control of queueing systems are major topics in the control, performance evaluation and optimization of production and service systems.

At my local supermarket, for instance, any customer that joins a queue of 4 or more customers gets his/her groceries for free. Of course, there are some constraints: at least one of the cashier facilities has to be unoccupied by a server and the customers in queue should be equally divided over the cashiers that are open (and perhaps there are some further rules, of which I am unaware). When  $\pi(n)$  denotes fraction of customers that ‘see upon arrival’ the system with  $n$  customers, the manager that controls the occupation of the cashier positions is focused on keeping  $\pi(4) + \pi(5) + \dots$ , i.e., the fraction of customers that see upon arrival a queue length exceeding 3, very small. In a sense, this is easy enough: just hire many cashiers. However, the cost of personnel may then outweigh the yearly average cost of paying the customer penalties. Thus, the manager’s problem becomes to plan and control the service capacity in such a way that both the penalties and the personnel cost are small.

Fast food restaurants also deal with many interesting queueing situations. Consider, for instance, the preparation of hamburgers. Typically, hamburgers are made-to-stock, in other words, they are prepared before the actual demand has arrived. Thus, hamburgers in stock can be interpreted as customers in queue waiting for service, where the service time is the time between the arrival of two customers that buy hamburgers. The hamburgers have a typical lifetime, and they have to be scrapped if they remain on the shelf longer than a specified amount of time. Thus, the waiting time of hamburgers has to be closely monitored. Of course, it is easy to achieve zero scrap cost, simply by keeping no stock at all. However, to prevent lost-sales, it is very important to maintain a certain amount of hamburgers in stock. Thus, the manager has to balance the scrap cost against the cost of lost sales. In more formal terms, the problem is to choose a policy to prepare hamburgers such that the cost of excess waiting time (scrap) is balanced against the cost of an empty queue (lost sales).

Service systems, such as hospitals, call centers, courts, and so on, have a certain capacity available to serve customers. The performance of such systems is, in part, measured by the total number of jobs processed per year and the fraction of jobs processed within a certain time frame between receiving and closing the job. Here the problem is to organize the capacity such that the sojourn time, i.e., the typical time a job spends in the system, does not exceed some threshold, and such that the system achieves a certain throughput, i.e., jobs served per year.

Clearly, all of the above systems can be seen as queueing systems that have to be monitored and controlled to achieve a certain performance. The performance analysis of such systems can, typically, be characterized by the following performance measures:

1. The fraction of time  $p(n)$  that the system contains  $n$  customers. In particular,  $1 - p(0)$ , i.e., the fraction of time the system contains jobs, is important, as this is a measure of the time-average occupancy of the servers, hence related to personnel cost.

2. The fraction of customers  $\pi(n)$  that ‘see upon arrival’ the system with  $n$  customers. This measure relates to customer perception and lost sales, i.e., fractions of arriving customers that do not enter the system.
3. The average, variance, and/or distribution of the waiting time.
4. The average, variance, and/or distribution of the number of customers in the system.

Here the system can be anything that is capable of holding jobs, such as a queue, the server(s), an entire court, patients waiting for an MRI scan in a hospital, and so on.

It is important to realize that a queueing system can, typically, be decomposed into *two subsystems*: the queue itself and the service system. Thus, we are concerned with three types of waiting: waiting in queue, i.e., *queueing time*, waiting while being in service, i.e., the *service time*, and the total waiting time in the system, i.e., the *sojourn time*.

### Organization

In these notes, we will be primarily concerned with making models of queueing systems such that we can compute or estimate the above-mentioned performance measures.

In Chapter 1 we construct queueing systems in discrete time and continuous time, and by implementing these models in code, we can simulate and analyze such systems. Simulation allows us to analyze many realistic queueing systems, while such systems are often (way) too hard to analyze by mathematical tools. Consider, for example, the service process at a check-in desk of an airline company. Business customers and economy customers are served by two separate queueing systems: the business customers are served by one server, say, while the economy class customers by three servers, say. What would happen to the sojourn time of the business customers if their server would be allowed to serve economy class customers when the business queue is empty? For the analysis of such complicated control policies, simulation appears to be the most natural approach.

Notwithstanding the power of simulation, it is often hard to obtain structural understanding into the behavior of queueing systems. Mathematical models, whether exact or approximate, are much more useful to help reason about and improve queueing systems, mainly because they offer insights into scaling laws, such as how the average waiting time depends on average service times or variability of such times. The aim of Chapter 3 is to apply Sakasegawa’s formula to understand how different production and service situations are affected by the system parameters such as service speed, batching rules, and outages. In passing we will use some general tools of probability theory, which proves useful for the sequel of the book.

In Chapter 5 we focus on exact models for single-station queueing systems, and provide the motivation that underlies Sakasegawa’s formula. The main idea here is to consider the *sample paths of a queueing process*, and assume that a typical sample path captures the ‘normal’ stochastic behavior of the system. This sample-path approach has two advantages. In the first place, many of the theoretical results follow from very concrete aspects of these sample paths. Second, the analysis of sample-paths carries over right away to simulation. In fact, simulation of a queueing system offers us one (or more) sample path(s), and based on such sample paths, we derive behavioral and statistical properties of the system. Thus, sample paths form a direct bridge between simulation on the one hand and mathematical analysis on the other.

Chapter 6 extends the models of the previous chapters to controlled queueing systems and queueing networks. The analysis of these systems requires a combination of material of

the previous chapters, but also other mathematical tools such as difference and differential equations, and non-negative matrices. As such, it also provides a stepping stone to the many-fold extensions of the theory to Markov decision theory, optimization, dynamic programming, and so on.

In our discussions we focus on obtaining an intuitive understanding of the analytical tools. For proofs and/or more extensive results we refer to the following books.

1. [Bolch et al. \[2006\]](#)
2. [El-Taha and Stidham Jr. \[1998\]](#)
3. [Tijms \[1994\]](#) and/or [Tijms \[2003\]](#)
4. [Capiński and Zastawniak \[2003\]](#)

### *Exercises*

The main text contains hardly any examples or derivations. Instead, the exercises provide the material to *illustrate* the material and help the reader study the material. Also, a substantial part of exercises consists of consistency checks, to show how new results reduce to old results; like this these exercises also provide relations between various parts of the text. Typically, such checks are trivial; however, the algebra can be quite difficult at times. Another part of the exercises form a set of questions any student should ask while studying the material (even though asking good questions is difficult). For this reason, the reader is urged to try to make as many exercises as possible. Finally, the intention of the exercises is not to be easy.

The companion document gives hints and solutions to all problems. The solutions spell out nearly every intermediate step. For most of you all, this detail is not necessary, but over the years I got many questions like: "how do you go from 'here' to 'there'?" As service, I then added such intermediate steps. As a consequence, the companion is quite extensive. The companion document also contains many additional simple exercises and old exam questions.

Exercises marked as 'not obligatory' are interesting, but (too) hard. I will not use this in an exam.

### *Acknowledgements*

I would like to acknowledge dr. J.W. Nieuwenhuis for our many discussions on queueing theory. To convince him about the more formal aspects, sample-path arguments proved very useful. Finally, I thank my students for submitting many improvements via github. It's very motivating to see a book like this turn into a joint piece of work.

## 0.2 PRELIMINARIES

Here is an overview of concepts you are supposed to have seen in earlier courses. We will use these concepts over and over in the rest of the book.

*Theory and Exercises*

We use the notation:

$$\begin{aligned} [x]^+ &= \max\{x, 0\}, \\ f(x-) &= \lim_{y \downarrow x} f(y), \\ f(x+) &= \lim_{y \uparrow x} f(y), \\ \mathbb{1}_A &= \begin{cases} 1, & \text{if } A \text{ is true,} \\ 0, & \text{if } A \text{ is false.} \end{cases} \end{aligned}$$

The last equation defines the *indicator function*.

We write  $f(h) = o(h)$  for a function  $f$  to say that  $f$  is such that  $f(h)/h \rightarrow 0$  as  $h \rightarrow 0$ . If we write  $f(h) = o(h)$  it is implicit that  $|h| \ll 1$ . We call this *small o notation*.

**0.2.1.** Let  $c$  be a constant (in  $\mathbb{R}$ ) and the functions  $f$  and  $g$  both of  $o(h)$ . Then show that (1)  $f(h) \rightarrow 0$  when  $h \rightarrow 0$ , (2)  $c \cdot f = o(h)$ , (3)  $f + g = o(h)$ , and (4)  $f \cdot g = o(h)$ .

You should know that:

$$(a + b)^n = \sum_{i=0}^n \binom{n}{i} a^{n-i} b^i, \quad (0.2.1a)$$

$$e^x = \lim_{n \rightarrow \infty} (1 + x/n)^n, \quad (0.2.1b)$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!}, \quad (0.2.1c)$$

$$\sum_{n=0}^N \alpha^n = \frac{1 - \alpha^{N+1}}{1 - \alpha}. \quad (0.2.1d)$$

**0.2.2.** Why is  $e^x = 1 + x + o(x)$ ?

**0.2.3.** What condition is necessary such that  $\sum_{n=0}^{\infty} \alpha^n = (1 - \alpha)^{-1}$ ?

You should know that for a non-negative, integer-valued random variable  $X$  with *probability mass function*  $f(k) = P(X = k)$ ,

$$X = \sum_{n=0}^{\infty} X \mathbb{1}_{X=n} = \sum_{n=0}^{\infty} n \mathbb{1}_{X=n}, \quad (0.2.2a)$$

$$E[X] = E\left[\sum_{n=0}^{\infty} n \mathbb{1}_{X=n}\right] = \sum_{n=0}^{\infty} n E[\mathbb{1}_{X=n}] = \sum_{n=0}^{\infty} n f(n), \quad (0.2.2b)$$

$$E[g(X)] = \sum_{n=0}^{\infty} g(n) f(n), \quad (0.2.2c)$$



For general random variables  $X$  and  $Y$ :

$$\mathbb{E}[\mathbb{1}_{X \leq x}] = \mathbb{P}(X \leq x), \quad (0.2.3)$$

$$\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2. \quad (0.2.4)$$

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y], \quad (0.2.5)$$

$$\mathbb{V}[X + Y] = \mathbb{V}[X] + \mathbb{V}[Y], \quad \text{if } X \text{ and } Y \text{ are independent.} \quad (0.2.6)$$

**0.2.4.** Why is (0.2.2a) true?

**0.2.5.** Define the survivor function of  $X$  as  $G(k) = \mathbb{P}(X > k)$ . Show that

$$G(k) = \sum_{m=0}^{\infty} \mathbb{1}_{m > k} f(m).$$

As you will see below, this idea makes the computation of certain expressions quite a bit easier.

**0.2.6.** Express the probability mass  $f(k)$  and the survivor function  $G(k)$  in terms of the distribution function  $F(k) = \mathbb{P}(X \leq k)$  of  $X$ .

**0.2.7.** Which of the following is true:  $G(k) = 1 - F(k)$ ,  $G(k) = 1 - F(k - 1)$ , or  $G(k) = 1 - F(k + 1)$ ?

**0.2.8.** Use indicator functions to prove that  $\mathbb{E}[X] = \sum_{k=0}^{\infty} G(k)$ .

**0.2.9.** Use indicator functions to prove that  $\sum_{i=0}^{\infty} iG(i) = \mathbb{E}[X^2]/2 - \mathbb{E}[X]/2$ .

Let  $X$  be a continuous non-negative random variable with distribution function  $F$ . We write

$$\mathbb{E}[X] = \int_0^{\infty} x \, dF(x)$$

for the expectation of  $X$ . Here  $dF(x)$  acts as a (sort of) shorthand for  $f(x)dx$ <sup>1</sup>. Recall that

$$\mathbb{E}[g(X)] = \int_0^{\infty} g(x) \, dF(x).$$

**0.2.10.** Use indicator functions to prove that  $\mathbb{E}[X] = \int_0^{\infty} x \, dF(x) = \int_0^{\infty} G(y) \, dy$ , where  $G(x) = 1 - F(x)$ .

You should be able to use indicator functions and integration by parts to show that  $\mathbb{E}[X^2] = 2 \int_0^{\infty} yG(y) \, dy$ , where  $G(x) = 1 - F(x)$ , provided the second moment exists.

**0.2.11.** Use indicator functions to prove that for a continuous non-negative random variable  $X$  with distribution function  $F$ ,  $\mathbb{E}[X^2] = \int_0^{\infty} x^2 \, dF(x) = 2 \int_0^{\infty} yG(y) \, dy$ , where  $G(x) = 1 - F(x)$ .

**0.2.12.** Show that  $\mathbb{E}[X^2]/2 = \int_0^{\infty} yG(y) \, dy$  for a continuous non-negative random variable  $X$  with survivor function  $G$ .

You should know that the *moment-generating function*  $M_X(s)$  of a random variable  $X$  and  $s \in \mathbb{R}$  sufficiently small is defined as:

$$M_X(s) = \mathbb{E}[e^{sX}]. \quad (0.2.7a)$$

<sup>1</sup> For the interested reader,  $\int x \, dF(x)$  is a Lebesgue-Stieltjes integral with respect to the measure induced by the distribution function  $F$ , see the literature for further background.

Moreover,  $M_X(s)$  uniquely characterizes the distribution of  $X$ . From this definition it follows that:

$$E[X] = M'_X(0) = \left. \frac{dM_X(s)}{ds} \right|_{s=0}, \quad (0.2.7b)$$

$$E[X^2] = M''_X(0), \quad (0.2.7c)$$

$$M_{X+Y}(s) = M_X(s) \cdot M_Y(s), \quad \text{if } X \text{ and } Y \text{ are independent.} \quad (0.2.7d)$$

**0.2.13.** What is the value of  $M_X(0)$ ?

To help you recall the concept of *conditional probability* consider the following question.

**0.2.14.** We have one gift to give to one out of three children. As we cannot divide the gift into parts, we decide to let 'fate decide'. That is, we choose a random number in the set  $\{1, 2, 3\}$ . The first child that guesses this number wins the gift. Show that the probability of winning the gift is the same for each child.

You should know that:

$$P(A|B) = \frac{P(AB)}{P(B)}, \quad \text{if } P(B) > 0, \quad (0.2.8a)$$

$$P(A) = \sum_{i=1}^n P(AB_i) = \sum_{i=1}^n P(A|B_i)P(B_i), \quad \text{if } A = \bigcup_{i=1}^n B_i \text{ and } P(B_i) > 0 \text{ for all } i. \quad (0.2.8b)$$

*Hints*

**h.0.2.6.** This exercise is just meant to become familiar with the notation.

**h.0.2.8.** Write  $\sum_{k=0}^{\infty} G(k) = \sum_{k=0}^{\infty} \sum_{m=k+1}^{\infty} P(X=m)$ , reverse the summations. Then realize that  $\sum_{k=0}^{\infty} \mathbb{1}_{k < m} = m$ . You should be aware that this sort of problem is just a regular probability theory problem, nothing fancy. We use/adapt the tools you learned in calculus to carry out 2D integrals (or in this case 2D summations).

**h.0.2.9.**  $\sum_{i=0}^{\infty} iG(i) = \sum_{n=0}^{\infty} P(X=n) \sum_{i=0}^{\infty} i \mathbb{1}_{n \geq i+1}$ , and reverse the summations.

**h.0.2.10.**  $E[X] = \int_0^{\infty} x dF(x) = \int_0^{\infty} \int_0^{\infty} \mathbb{1}_{y \leq x} dy dF(x)$ .

**h.0.2.11.**  $\int_0^{\infty} yG(y)dy = \int_0^{\infty} y \int_0^{\infty} \mathbb{1}_{y \leq x} f(x) dx dy$ .

**h.0.2.14.** For the second child, condition on the event that the first does not choose the right number. Use the definition of conditional probability:  $P(A|B) = P(AB)/P(B)$  provided  $P(B) > 0$ .

*Solutions*

**s.0.2.1.** In fact (1) is trivial:  $|f(h)| \leq |f(h)/h|$  when  $|h| < 1$ . But it is given that the right-hand side goes to zero. For (2) and (3):

$$\begin{aligned} \lim_{h \rightarrow 0} \frac{cf(h)}{h} &= c \lim_{h \rightarrow 0} \frac{f(h)}{h} = 0, \text{ as } f = o(h), \\ \lim_{h \rightarrow 0} \frac{f(h) + g(h)}{h} &= \lim_{h \rightarrow 0} \frac{f(h)}{h} + \lim_{h \rightarrow 0} \frac{g(h)}{h} = 0. \end{aligned}$$

For (4), use the Algebraic Limit Theorem,

$$\begin{aligned}\lim_{h \rightarrow 0} \frac{f(h)g(h)}{h} &= \lim_{h \rightarrow 0} h \frac{f(h)}{h} \frac{g(h)}{h} \\ &= \lim_{h \rightarrow 0} h \lim_{h \rightarrow 0} \frac{f(h)}{h} \lim_{h \rightarrow 0} \frac{g(h)}{h} \\ &= 0 \cdot 0 \cdot 0 = 0.\end{aligned}$$

**s.0.2.2.** When  $|x| \ll 1$ , the terms with  $n \geq 2$  in (0.2.1c) are  $x^n = o(x)$ . Then applying  $x^n + x^m = o(x)$  to the Taylor series gives the result.

**s.0.2.3.** Assuming that  $\alpha$  is real number, we need that  $\alpha \in (-1, 1)$

**s.0.2.4.** To see (0.2.2a), note first that  $X \mathbb{1}_{X=n} = n \mathbb{1}_{X=n}$  because  $X = n$  when  $\mathbb{1}_{X=n} = 1$ , and second that  $\sum_{n=0}^{\infty} \mathbb{1}_{X=n} = 1$ , since  $X$  takes one of the values in  $\mathbb{N}$ , and events  $\{X = n\}$  and  $\{X = m\}$  are non-overlapping when  $n \neq m$ .

**s.0.2.5.** This is just rewriting the definition:

$$G(k) = P(X > k) = \sum_{m=k+1}^{\infty} P(X = m) = \sum_{m=k+1}^{\infty} f(m) = \sum_{m=0}^{\infty} \mathbb{1}_{m>k} f(m).$$

**s.0.2.6.**

$$\begin{aligned}f(k) &= P(X = k) = P(X \leq k) - P(X \leq k-1) = F(k) - F(k-1), \\ G(k) &= P(X > k) = 1 - P(X \leq k) = 1 - F(k).\end{aligned}$$

**s.0.2.7.**  $G(k) = 1 - F(k) = 1 - P(X \leq k) = P(X > k)$ . It is all too easy to make, so called, off-by-one errors, such as in the three alternatives above. I nearly always check simple cases to prevent such simple mistakes. I advise you to acquire the same habit.

**s.0.2.8.** Observe first that  $\sum_{k=0}^{\infty} \mathbb{1}_{m>k} = m$ , since  $\mathbb{1}_{m>k} = 1$  if  $k < m$  and  $\mathbb{1}_{m>k} = 0$  if  $k \geq m$ . With this,

$$\begin{aligned}\sum_{k=0}^{\infty} G(k) &= \sum_{k=0}^{\infty} P(X > k) = \sum_{k=0}^{\infty} \sum_{m=k+1}^{\infty} P(X = m) \\ &= \sum_{k=0}^{\infty} \sum_{m=0}^{\infty} \mathbb{1}_{m>k} P(X = m) = \sum_{m=0}^{\infty} \sum_{k=0}^{\infty} \mathbb{1}_{m>k} P(X = m) \\ &= \sum_{m=0}^{\infty} m P(X = m) = E[X].\end{aligned}$$

In case you are interested in mathematical justifications: the interchange of the two summations is allowed by Tonelli's theorem because the summands are all positive. (Interchanging the order of summations or integration is not always allowed because the results can be different when part of the integrand is negative. Check Fubini's theorem for more on this if you are interested.)

**s.0.2.9.**

$$\begin{aligned}\sum_{i=0}^{\infty} i G(i) &= \sum_{i=0}^{\infty} i \sum_{n=i+1}^{\infty} P(X = n) = \sum_{n=0}^{\infty} P(X = n) \sum_{i=0}^{\infty} i \mathbb{1}_{n \geq i+1} \\ &= \sum_{n=0}^{\infty} P(X = n) \sum_{i=0}^{n-1} i = \sum_{n=0}^{\infty} P(X = n) \frac{(n-1)n}{2} \\ &= \sum_{n=0}^{\infty} \frac{n^2}{2} P(X = n) - \frac{E[X]}{2} = \frac{E[X^2]}{2} - \frac{E[X]}{2}.\end{aligned}$$

**s.0.2.10.**

$$\begin{aligned}
E[X] &= \int_0^\infty x \, dF(x) = \int_0^\infty \int_0^x dy \, dF(x) \\
&= \int_0^\infty \int_0^\infty \mathbb{1}_{y \leq x} dy \, dF(x) = \int_0^\infty \int_0^\infty \mathbb{1}_{y \leq x} dF(x) dy \\
&= \int_0^\infty \int_y^\infty dF(x) dy = \int_0^\infty G(y) dy.
\end{aligned}$$

**s.0.2.11.**

$$\begin{aligned}
\int_0^\infty yG(y) dy &= \int_0^\infty y \int_y^\infty f(x) dx dy = \int_0^\infty y \int_0^\infty \mathbb{1}_{y \leq x} f(x) dx dy \\
&= \int_0^\infty f(x) \int_0^\infty y \mathbb{1}_{y \leq x} dy dx = \int_0^\infty f(x) \int_0^x y dy dx \\
&= \int_0^\infty f(x) \frac{x^2}{2} dx = \frac{E[X^2]}{2}.
\end{aligned}$$

**s.0.2.12.** Use integration by parts.

$$\int_0^\infty yG(y) dy = \frac{y^2}{2}G(y)\Big|_0^\infty - \int_0^\infty \frac{y^2}{2}g(y) dy = \int_0^\infty \frac{y^2}{2}f(y) dy = \frac{E[X^2]}{2}, \quad (0.2.9)$$

since  $g(y) = G'(y) = -F'(y) = -f(y)$ . Note that we used  $\frac{y^2}{2}G(y)\Big|_0^\infty = 0 - 0 = 0$ , which follows from our assumption that  $E[X^2]$  exists, implying that  $\lim_{y \rightarrow \infty} y^2G(y) = 0$ .

**s.0.2.13.**  $M_X(0) = E[e^{0X}] = E[e^0] = E[1] = 1$ .

**s.0.2.14.** The probability that the first child to guess also wins is  $1/3$ . What is the probability for child number two? Well, for him/her to win, it is necessary that child one does not win and that child two guesses the right number of the remaining numbers. Assume, without loss of generality that child 1 chooses 3 and that this is not the right number. Then

$$\begin{aligned}
&P(\text{Child 2 wins}) \\
&= P(\text{Child 2 guesses the right number and child 1 does not win}) \\
&= P(\text{Child 2 guesses the right number} \mid \text{child 1 does not win}) \cdot P(\text{Child 1 does not win}) \\
&= P(\text{Child 2 makes the right guess in the set } \{1, 2\}) \cdot \frac{2}{3} \\
&= \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}.
\end{aligned}$$

Similar conditional reasoning gives that child 3 wins with probability  $1/3$ .

## CONSTRUCTION AND SIMULATION OF QUEUEING SYSTEMS

---

The first step to analyze a queueing system is to model it. And for this, there is often not a better start than to build a simulation model. Thus, the aim of this chapter is to teach you how to construct and simulate queueing processes.

In Section 1.2 we build discrete-time models of queueing systems, which means that we use the number of jobs that arrive and can be served in a period to construct the queueing process. Such a period can be an hour, or a day; in fact, any amount of time that makes sense in the context in which the model will be used. Typically we model the number of arrivals and potential services as random variables, and in many practical settings it is reasonable to take the number of arrivals in a period as Poisson distributed. This being the case, we consider the Poisson distribution in Section 1.1, and once we have an understanding of this, we can use random number generators to generate (Poisson distributed) random numbers of arrivals and services to drive the simulator.

In Section 1.4 we focus on constructing queueing processes in continuous time. In this setting, the inter-arrival times and service times of individual jobs become of importance, and then exponentially distributed random variables play a fundamental role. We therefore discuss the properties of the exponential distribution in Section 1.3. We also mention the interesting and close relationship between the exponential distribution and the Poisson distribution.

As will become apparent, both types of constructing queueing processes, the discrete-time or continuous-time models, are easy to implement as computer programs. We include a large number of exercises to show you the astonishing diversity of queueing systems that can be analyzed by simulation. In passing, we develop a number of performance measures to provide insight into the (transient and long-run average) behavior of queueing processes.

We expect you to *know all topics* summarized in Section 0.2; we use these extensively in Section 1.1 and Section 1.3, as well as in any other section that introduces some theory.

### 1.1 POISSON DISTRIBUTION

In this section, we provide motivation for the use of the Poisson process as an arrival process of customers or jobs at a shop, service station, or machine to receive service. In the exercises we derive numerous properties of this exceedingly important distribution; in the rest of the book we will use these results time and again.

#### *Theory and Exercises*

Consider a stream of customers that enter a shop over time. Let us write  $N(t)$  for the number of customers that enter during the time interval  $[0, t]$  and  $N(s, t) = N(t) - N(s)$  for the number that arrive in the time period  $(s, t]$ . Clearly, as we do not know in advance how many customers will enter, we model the set  $\{N(t), t \geq 0\}$  as a family of random variables.

Our first assumption is that the rate at which customers enter stays constant over time. Then it is reasonable to assume that the expected number of arrivals is proportional to the

length of the interval. Hence, it is reasonable to assume that there exists some constant  $\lambda$  such that

$$E[N(s, t)] = \lambda(t - s). \quad (1.1.1)$$

The constant  $\lambda$  is called the *arrival rate* of the arrival process.

The second assumption is that the process  $N_\lambda = \{N(t), t \geq 0\}$  has *stationary and independent increments*. Stationarity means that the distributions of the number of arrivals are the same for all intervals of equal length, that is,  $N(s, t]$  has the same distribution as  $N(u, v]$  if  $t - s = v - u$ . Independence means, roughly speaking, that knowing that  $N(s, t] = n$ , does not help to make any predictions about the value of  $N(u, v]$  if the intervals  $(s, t]$  and  $(u, v]$  do not overlap.

To find the distribution of  $N(t)$  for some given  $t$ , let us split the interval  $[0, t]$  into  $n$  sub-intervals, all of equal length, and ask: ‘What is the probability that a customer arrives in some given sub-interval?’ By our first assumption, the arrival rate is constant over time. Therefore, the probability  $p$  of an arrival in each interval should be constant. Moreover, if the time intervals are very small, we can safely neglect the probability that two or more customers arrive in one interval.

As a consequence, then, we can model the occurrence of an arrival in some period  $i$  as a Bernoulli distributed random variable  $B_i$ . We assume that the  $B_i, i = 1, \dots$  are independent and identically distributed (*i.i.d.*) as the common random variable  $B$  such that  $p = P(B = 1)$  and  $1 - p = P(B = 0)$ . The total number of arrivals  $N_n(t)$  that occur in  $n$  intervals is then *binomially distributed*, i.e.,

$$P(N_n(t) = k) = \binom{n}{k} p^k (1 - p)^{n-k}. \quad (1.1.2)$$

If we take  $n \rightarrow \infty$ ,  $p \rightarrow 0$  such that  $np = \lambda t$ , then  $N_n(t)$  converges (in distribution) to a *Poisson distributed* random variable  $N(t)$ , i.e.,

$$P(N(t) = k) = e^{-\lambda t} \frac{(\lambda t)^k}{k!}, \quad (1.1.3)$$

and then we write  $N(t) \sim P(\lambda t)$ .

**1.1.1.** Show that  $E[N_n(t)] = \sum_{i=1}^n E[B_i] = np$ . Conclude that we need to choose  $p = \lambda t/n$  if we want that  $E[N_n(t)] = E[N(t)]$ .

**1.1.2.** What is the difference between  $N_n(t)$  and  $N(t)$ ?

**1.1.3.** Show that the binomial distribution in (1.1.2) converges to the Poisson distribution, i.e.,

$$\lim_{n \rightarrow \infty} \binom{n}{k} \left( \frac{\lambda t}{n} \right)^k \left( 1 - \frac{\lambda t}{n} \right)^{n-k} = e^{-\lambda t} \frac{(\lambda t)^k}{k!},$$

if  $n \rightarrow \infty$ ,  $p \rightarrow 0$  such that  $np = \lambda t$ .

We call the process  $N_\lambda = \{N(t)\}$  a *Poisson process* with rate  $\lambda$  when  $N_\lambda$  is stationary, has independent increments, and its elements  $N(t) \sim P(\lambda t)$  for all  $t$ . Observe that the process  $N_\lambda$  is a much more complicated object than a Poisson distributed random variable. The process is an uncountable set of random variables indexed by  $t \in \mathbb{R}^+$ , not just *one* random variable.

In the remainder of this section we derive a number of properties of the Poisson process that we will use time and again.

**1.1.4.** Show that

$$P(N(t+h) = n | N(t) = n) = 1 - \lambda h + o(h)$$

when  $N(t) \sim P(\lambda t)$  and  $h$  is small.

**1.1.5.** Show that

$$P(N(t+h) = n+1 | N(t) = n) = \lambda h + o(h)$$

when  $N(t) \sim P(\lambda t)$  and  $h$  is small.

**1.1.6.** Show that if  $N(t) \sim P(\lambda t)$ , we have for small  $h$ ,

$$P(N(t+h) \geq n+2 | N(t) = n) = o(h).$$

**1.1.7.** Show that for a Poisson process  $N_\lambda$ ,

$$P(N(s) = 1 | N(t) = 1) = \frac{s}{t},$$

if  $s \in [0, t]$ . Thus, if you know that an arrival occurred during  $[0, t]$ , the arrival is distributed uniformly on the interval  $[0, t]$ . Note that this probability is independent of  $\lambda$ .

**1.1.8.** Show that if  $N(t) \sim P(\lambda t)$ , the expected number of arrivals during  $[0, t]$  is

$$E[N(t)] = \lambda t.$$

**1.1.9.** Show that  $E[(N(t))^2] = \lambda^2 t^2 + \lambda t$  if  $N(t) \sim P(\lambda t)$ .

**1.1.10.** Show that if  $N(t) \sim P(\lambda t)$ , the variance of the number of arrivals during  $[0, t]$  is  $V[N(t)] = \lambda t$ .

**1.1.11.** Show that the moment-generating function of the random variable  $N(t) \sim P(\lambda t)$  is

$$M_{N(t)}(s) = \exp(\lambda t(e^s - 1)).$$

**1.1.12.** Use the moment-generating function of  $N(t) \sim P(\lambda t)$  to compute  $E[N(t)]$  and  $V[N(t)]$ .

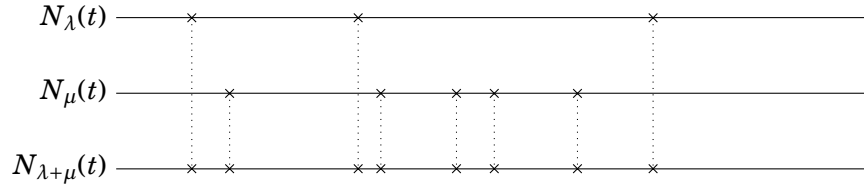
Define the *square coefficient of variation* (SCV) of a random variable  $X$  as

$$C^2 = \frac{V[X]}{(E[X])^2}. \quad (1.1.4)$$

As will become clear later, the SCV is a very important concept in queueing theory. Memorize it as a measure of *relative variability*.

**1.1.13.** Show that the SCV of  $N(t) \sim P(\lambda t)$  is equal to  $1/(\lambda t)$ . What does this mean for  $t$  large?

*Merging* Poisson processes occurs often in practice. We have two Poisson processes, for instance, the arrival processes  $N_\lambda$  of men and  $N_\mu$  of women at a shop. In the figure below, each cross represents an arrival; in the upper line it corresponds to a man, in the middle line to a woman, and in the lower line to an arrival of a general customer at the shop. Thus, the shop ‘sees’ the superposition of these two arrival processes. In fact, this merged process  $N_{\lambda+\mu}$  is also a Poisson process with rate  $\lambda + \mu$ .



**1.1.14.** If the Poisson arrival processes  $N_\lambda$  and  $N_\mu$  are independent, show with a conditioning argument that  $N_\lambda + N_\mu$  is a Poisson process with rate  $\lambda + \mu$ .

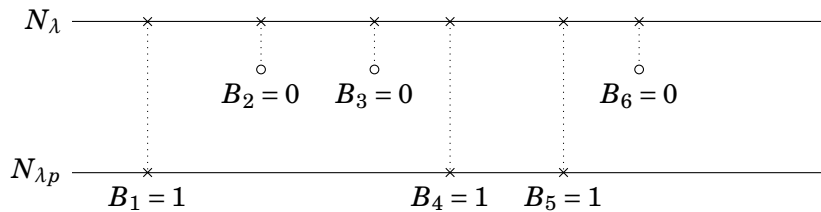
**1.1.15.** If the Poisson arrival processes  $N_\lambda$  and  $N_\mu$  are independent, use moment-generating functions to show that  $N_\lambda + N_\mu$  is a Poisson process with rate  $\lambda + \mu$ .

**1.1.16.** If the Poisson arrival processes  $N_\lambda$  and  $N_\mu$  are independent, show that

$$P(N_\lambda(h) = 1 | N_\lambda(h) + N_\mu(h) = 1) = \frac{\lambda}{\lambda + \mu}.$$

Note that the right-hand side does not depend on  $h$ , hence it holds for any time  $h$ , whether it is small or not.

Besides merging Poisson streams, we can also consider the concept of *splitting*, or *thinning*, a stream into sub-streams, as follows. Model the stream of people passing by a shop as a Poisson process  $N_\lambda$ . In the figure below these arrivals are marked as crosses at the upper line. With probability  $p$  a person decides, independent of anything else, to enter the shop; the crosses at the lower line are the customers that enter the shop. In the figure, the Bernoulli random variable  $B_1 = 1$  so that the first passerby enters the shop; the second passerby does not enter as  $B_2 = 0$ , and so on.



**1.1.17.** Show with conditioning that thinning the Poisson process  $N_\lambda$  by means of Bernoulli random variables with success probability  $p$  results in a Poisson process  $N_{\lambda p}$ .

**1.1.18.** Show with moment-generating functions that thinning the Poisson process  $N_\lambda$  by means of Bernoulli random variables with success probability  $p$  results in a Poisson process  $N_{\lambda p}$ .

The concepts of merging and thinning are useful to analyze queueing networks. Suppose the departure stream of a machine splits into two sub-streams, e.g., a fraction  $p$  of the jobs moves on to another machine and the rest  $(1 - p)$  of the jobs leaves the system. Then we can model the arrival stream at the second machine as a thinned stream (with probability  $p$ ) of the departures of the first machine. Merging occurs where the output streams of various stations arrive at another station.

**1.1.19.** Use moment-generating functions to prove that  $N_n(t)$  converges to  $N$ , that is, the right-hand side in (1.1.2) converges to the Poisson distribution (1.1.3) when  $n \rightarrow \infty, p \rightarrow 0$  such that  $pn = \lambda t$  remains constant.



*Hints*

**h.1.1.1.** Use that  $E[X + Y] = E[X] + E[Y]$ .

**h.1.1.3.** First find  $p$ ,  $n$ ,  $\lambda$  and  $t$  such that the rate at which an event occurs in both processes are the same. Then consider the binomial distribution and use the standard limit  $(1 - x/n)^n \rightarrow e^{-x}$  as  $n \rightarrow \infty$ .

**h.1.1.4.** Use the definition of the conditional probability and small  $o$  notation.

**h.1.1.5.** Use [1.1.4].

**h.1.1.6.** Use that  $\sum_{i=2}^{\infty} x^i/i! = \sum_{i=0}^{\infty} x^i/i! - x - 1 = e^x - x - 1$ .

**h.1.1.7.** Observe that

$$\mathbb{1}_{N(0,s]+N(s,t)=1} \mathbb{1}_{N(0,s)=1} = \mathbb{1}_{1+N(s,t)=1} \mathbb{1}_{N(0,s)=1} = \mathbb{1}_{N(s,t)=0} \mathbb{1}_{N(0,s)=1}.$$

Use independence and (0.2.3).

**h.1.1.8.** Use (0.2.2c). Note that the term with  $n = 0$  does not contribute in the following summation

$$\sum_{n=0}^{\infty} n \frac{\lambda^n}{n!} = \sum_{n=1}^{\infty} n \frac{\lambda^n}{n!} = \sum_{n=1}^{\infty} \frac{\lambda^n}{(n-1)!} = \lambda \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} = \lambda e^{\lambda},$$

where we apply a change of notation in the second to last step.

**h.1.1.10.** Use (0.2.4), [1.1.9], and [1.1.8].

**h.1.1.11.** Use (0.2.2c) with  $f(x) = e^{sx}$ .

**h.1.1.12.** Use (0.2.7b) and (0.2.7c).

**h.1.1.14.** Use sets  $\{N_{\lambda}(t) = i\}$  to decompose  $\{N_{\lambda}(t) + N_{\mu}(t) = n\}$ . With this observe that

$$\mathbb{1}_{N_{\lambda}(t)+N_{\mu}(t)=n} = \sum_{i=0}^n \mathbb{1}_{N_{\lambda}(t)=i, N_{\mu}(t)=n-i}.$$

Take expectations left and right, use (0.2.3), and independence of  $N_{\lambda}$  and  $N_{\mu}$ . Near the end of the computation, use (0.2.1a).

**h.1.1.15.** Use (0.2.7d) and (0.2.7a).

**h.1.1.16.** Use the standard formula for conditional probability and that  $N_{\lambda}(t) + N_{\mu}(t) \sim P((\lambda + \mu)t)$ . Interpret the result.

**h.1.1.17.** Suppose that  $N_1$  is the thinned stream, and  $N$  the original stream. Condition on the total number of arrivals  $N(t) = n$  up to time  $t$ . Then, realize that the probability that a person is of type 1 is  $p$ . Hence when you consider  $n$  people in total, the number  $N_1(t)$  of type 1 people is binomially distributed. Thus, given that  $n$  people arrived, the probability of  $k$  ‘successes’ (i.e., arrivals of type 1), is

$$P(N_1(t) = k | N(t) = n) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Use (0.2.8b) to decompose the  $\{N_1 = k\}$ , and (0.2.1c) at the end.

**h.1.1.18.** Dropping the dependence of  $N$  on  $t$  for the moment for notational convenience, consider the random variable

$$Y = \sum_{i=1}^N Z_i,$$

with  $N \sim P(\lambda)$  and  $Z_i \sim B(p)$ . Show that the moment-generating function of  $Y$  is equal to the moment-generating function of a Poisson random variable with parameter  $\lambda p$ .

**h.1.1.19.** Solve [1.1.11] and [1.1.18] first. Perhaps [1.1.3] is also useful.

*Solutions*

**s.1.1.1.**

$$\mathbb{E}[N_n(t)] = \mathbb{E}\left[\sum_{i=1}^n B_i\right] = \sum_{i=1}^n \mathbb{E}[B_i] = n \mathbb{E}[B_i] = np.$$

By (1.1.1)  $\mathbb{E}[N(t)] = \lambda t$ . Now we want the expectations of  $N_n(t)$  and  $N(t)$  to be the same, thus,  $np = \lambda t$ .

**s.1.1.2.**  $N_n(t)$  is a binomially distributed random variable with parameters  $n$  and  $p$ . The maximum value of  $N_n(t)$  is  $n$ . The random variable  $N(t)$  models the number of arrivals that can occur during  $[0, t]$ . As such it is not necessarily bounded by  $n$ . Thus,  $N_n(t)$  and  $N(t)$  cannot represent the same random variable.

**s.1.1.3.**

$$\begin{aligned} \binom{n}{k} \left(\frac{\lambda t}{n}\right)^k \left(1 - \frac{\lambda t}{n}\right)^{n-k} &= \frac{n!}{k!(n-k)!} \left(\frac{\lambda t}{n} \frac{n}{n-\lambda t}\right)^k \left(1 - \frac{\lambda t}{n}\right)^n \\ &= \frac{(\lambda t)^k}{k!} \left(\frac{n}{n-\lambda t}\right)^k \frac{n!}{n^k(n-k)!} \left(1 - \frac{\lambda t}{n}\right)^n \\ &= \frac{(\lambda t)^k}{k!} \left(\frac{n}{n-\lambda t}\right)^k \frac{n}{n} \frac{n-1}{n} \cdots \frac{n-k+1}{n} \left(1 - \frac{\lambda t}{n}\right)^n. \end{aligned}$$

Observe now that, as  $\lambda t$  is finite,  $n/(n-\lambda t) \rightarrow 1$  as  $n \rightarrow \infty$ . Also for any finite  $k$ ,  $(n-k)/n \rightarrow 1$ . Finally, use (0.2.1b) to see that  $\left(1 - \frac{\lambda t}{n}\right)^n \rightarrow e^{-\lambda t}$ .

**s.1.1.4.** Write  $N(s, t]$  for the number of arrivals in the interval  $(s, t]$ . First we make a few simple observations:  $N(t+h) = N(t) + N(t, t+h]$ , hence

$$\begin{aligned} \mathbb{1}_{N(t+h)=n, N(t)=n} &= \mathbb{1}_{N(t)+N(t, t+h]=n, N(t)=n} = \\ &= \mathbb{1}_{N(t, t+h]=0, N(t)=n}. \end{aligned}$$

Thus,

$$\begin{aligned} P(N(t+h) = n | N(t) = n) &= \frac{P(N(t+h) = n, N(t) = n)}{P(N(t) = n)} \\ &= \frac{P(N(t, t+h] = 0, N(t) = n)}{P(N(t) = n)} \\ &= \frac{P(N(t, t+h] = 0) P(N(t) = n)}{P(N(t) = n)} \quad (\text{independence}) \end{aligned}$$

$$\begin{aligned}
&= P(N(t, t+h] = 0) \\
&= P(N(0, h] = 0) \quad (\text{stationarity}) \\
&= e^{-\lambda h} (\lambda h)^0 / 0! \\
&= e^{-\lambda h} = 1 - \lambda h + o(h).
\end{aligned}$$

**s.1.1.5.**

$$\begin{aligned}
P(N(t+h) = n+1 | N(t) = n) &= P(N(t+h) = n+1, N(t) = n) / P(N(t) = n) \\
&= P(N(t, t+h] = 1) = e^{-\lambda h} (\lambda h)^1 / 1! \\
&= (1 - \lambda h + o(h)) \lambda h = \lambda h - \lambda^2 h^2 + o(h) \\
&= \lambda h + o(h).
\end{aligned}$$

**s.1.1.6.**

$$\begin{aligned}
P(N(t+h) \geq n+2 | N(t) = n) &= P(N(t, t+h] \geq 2) \\
&= e^{-\lambda h} \sum_{i=2}^{\infty} \frac{(\lambda h)^i}{i!} = e^{-\lambda h} \left( \sum_{i=0}^{\infty} \frac{(\lambda h)^i}{i!} - \lambda h - 1 \right) \\
&= e^{-\lambda h} (e^{\lambda h} - 1 - \lambda h) = 1 - e^{-\lambda h} (1 + \lambda h) \\
&= 1 - (1 - \lambda h + o(h))(1 + \lambda h) = 1 - (1 - \lambda^2 h^2 + o(h)) \\
&= \lambda^2 h^2 + o(h) = o(h).
\end{aligned}$$

We can also use the results of the previous parts to see that

$$\begin{aligned}
P(N(t+h) \geq n+2 | N(t) = n) &= P(N(t, t+h] \geq 2) = 1 - P(N(t, t+h] < 2) \\
&= 1 - P(N(t, t+h] = 0) - P(N(t, t+h] = 1) \\
&= 1 - (1 - \lambda h + o(h)) - (\lambda h + o(h)) \\
&= o(h).
\end{aligned}$$

**s.1.1.7.** From the hint,

$$\begin{aligned}
P(N(0, s] = 1 | N(0, t] = 1) &= \frac{P(N(0, s] = 1, N(0, t] = 1)}{P(N(0, t] = 1)} \\
&= \frac{P(N(0, s] = 1, N(s, t] = 0)}{P(N(0, t] = 1)} \\
&= \frac{P(N(0, s] = 1) P(N(s, t] = 0)}{P(N(0, t] = 1)} \\
&= \frac{\lambda s e^{-\lambda s} e^{-\lambda(t-s)}}{\lambda t e^{-\lambda t}} = \frac{s}{t}.
\end{aligned}$$

**s.1.1.8.** When a random variable  $N$  is Poisson distributed with parameter  $\lambda t$ ,

$$\begin{aligned}
E[N] &= \sum_{n=0}^{\infty} n e^{-\lambda t} \frac{(\lambda t)^n}{n!} \\
&= \sum_{n=1}^{\infty} n e^{-\lambda t} \frac{(\lambda t)^n}{n!} \\
&= e^{-\lambda t} \lambda t \sum_{n=1}^{\infty} \frac{(\lambda t)^{n-1}}{(n-1)!}
\end{aligned}$$

$$\begin{aligned}
&= e^{-\lambda t} \lambda t \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} \\
&= e^{-\lambda t} \lambda t e^{\lambda t} \\
&= \lambda t.
\end{aligned}$$

**s.1.1.9.**

$$\begin{aligned}
E[N^2] &= \sum_{n=0}^{\infty} n^2 e^{-\lambda t} \frac{(\lambda t)^n}{n!} \\
&= e^{-\lambda t} \sum_{n=1}^{\infty} n \frac{(\lambda t)^n}{(n-1)!} \\
&= e^{-\lambda t} \sum_{n=0}^{\infty} (n+1) \frac{(\lambda t)^{n+1}}{n!} \\
&= e^{-\lambda t} \lambda t \sum_{n=0}^{\infty} n \frac{(\lambda t)^n}{n!} + e^{-\lambda t} \lambda t \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} \\
&\quad \text{now use the hint to simplify the first summation} \\
&= (\lambda t)^2 + \lambda t.
\end{aligned}$$

**s.1.1.10.**  $V[N] = E[N^2] - (E[N])^2 = (\lambda t)^2 + \lambda t - (\lambda t)^2 = \lambda t.$

**s.1.1.11.** Since  $N(t)$  is Poisson distributed with parameter  $\lambda t$ ,

$$\begin{aligned}
M_{N(t)}(s) &= E[e^{sN(t)}] \\
&= \sum_{k=0}^{\infty} e^{sk} P(N(t) = k) \\
&= \sum_{k=0}^{\infty} e^{sk} \frac{(\lambda t)^k}{k!} e^{-\lambda t} \\
&= e^{-\lambda t} \sum_{k=0}^{\infty} \frac{(e^s \lambda t)^k}{k!} \\
&= \exp(-\lambda t + e^s \lambda t) = \exp(\lambda t(e^s - 1)).
\end{aligned}$$

**s.1.1.12.** Using the expression for the moment-generating function of [1.1.11],

$$M'_{N(t)}(s) = \lambda t e^s \exp(\lambda t(e^s - 1)).$$

Hence  $E[N(t)] = M'_{N(t)}(0) = \lambda t$ . Next,

$$M''_{N(t)}(s) = (\lambda t e^s + (\lambda t e^s)^2) \exp(\lambda t(e^s - 1)),$$

hence  $E[(N(t))^2] = M''(0) = \lambda t + (\lambda t)^2$ . And thus,

$$V[N(t)] = E[(N(t))^2] - (E[N(t)])^2 = \lambda t + (\lambda t)^2 - (\lambda t)^2 = \lambda t.$$

**s.1.1.13.**

$$SCV = \frac{V[N(t)]}{(E[N(t)])^2} = \frac{\lambda t}{(\lambda t)^2} = \frac{1}{\lambda t}.$$

The relative variability of the Poisson process goes down as  $t \rightarrow \infty$ .

**s.1.1.14.**

$$\begin{aligned}
 P(N_\lambda(t) + N_\mu(t) = n) &= \sum_{i=0}^n P(N_\mu(t) = n - i) P(N_\lambda(t) = i) \\
 &= \sum_{i=0}^n \frac{(\mu t)^{n-i}}{(n-i)!} \frac{(\lambda t)^i}{i!} e^{-(\mu+\lambda)t} \\
 &= e^{-(\mu+\lambda)t} \sum_{i=0}^n \frac{(\mu t)^{n-i}}{(n-i)!} \frac{(\lambda t)^i}{i!} \\
 &= e^{-(\mu+\lambda)t} \frac{1}{n!} \sum_{i=0}^n \binom{n}{i} (\mu t)^{n-i} (\lambda t)^i \quad (\text{binomial formula}) \\
 &= \frac{((\mu + \lambda)t)^n}{n!} e^{-(\mu+\lambda)t}.
 \end{aligned}$$

**s.1.1.15.**

$$\begin{aligned}
 M_{N_\lambda(t)+N_\mu(t)}(s) &= M_{N_\lambda(t)}(s) \cdot M_{N_\mu(t)}(s) \\
 &= \exp(\lambda t(e^s - 1)) \cdot \exp(\mu t(e^s - 1)) \\
 &= \exp((\lambda + \mu)t(e^s - 1)).
 \end{aligned}$$

The last expression is the moment-generating function of a Poisson random variable with parameter  $(\lambda + \mu)t$ .

**s.1.1.16.** With the above:

$$\begin{aligned}
 P(N_\lambda(h) = 1 | N_\lambda(h) + N_\mu(h) = 1) &= \frac{P(N_\lambda(h) = 1, N_\lambda(h) + N_\mu(h) = 1)}{P(N_\lambda(h) + N_\mu(h) = 1)} \\
 &= \frac{P(N_\lambda(h) = 1, N_\mu(h) = 0)}{P(N_{\lambda+\mu}(h) = 1)} \\
 &= \frac{P(N_\lambda(h) = 1)P(N_\mu(h) = 0)}{P(N_{\lambda+\mu}(h) = 1)} \\
 &= \frac{\lambda h \exp(-\lambda h) \exp(-\mu h)}{((\lambda + \mu)h) \exp(-(\lambda + \mu)h)} \\
 &= \frac{\lambda h \exp(-(\lambda + \mu)h)}{((\lambda + \mu)h) \exp(-(\lambda + \mu)h)} \\
 &= \frac{\lambda}{\lambda + \mu}.
 \end{aligned}$$

Given that a customer arrived in  $[0, t]$ , the probability that it is of the first type is  $\lambda/(\lambda + \mu)$ .

**s.1.1.17.**

$$\begin{aligned}
 P(N_1(t) = k) &= \sum_{n=k}^{\infty} P(N_1(t) = k, N(t) = n) = \sum_{n=k}^{\infty} P(N_1(t) = k | N(t) = n) P(N(t) = n) \\
 &= \sum_{n=k}^{\infty} P(N_1(t) = k | N(t) = n) e^{-\lambda t} \frac{(\lambda t)^n}{n!} \\
 &= \sum_{n=k}^{\infty} \binom{n}{k} p^k (1-p)^{n-k} e^{-\lambda t} \frac{(\lambda t)^n}{n!}, \quad \text{by the hint}
 \end{aligned}$$

$$\begin{aligned}
&= e^{-\lambda t} \sum_{n=k}^{\infty} \frac{p^k (1-p)^{n-k}}{k!(n-k)!} (\lambda t)^n = e^{-\lambda t} \frac{(\lambda t p)^k}{k!} \sum_{n=k}^{\infty} \frac{(\lambda t (1-p))^{n-k}}{(n-k)!} \\
&= e^{-\lambda t} \frac{(\lambda t p)^k}{k!} \sum_{n=0}^{\infty} \frac{(\lambda t (1-p))^n}{n!} = e^{-\lambda t} \frac{(\lambda t p)^k}{k!} e^{\lambda t (1-p)} \\
&= e^{-\lambda t p} \frac{(\lambda t p)^k}{k!}.
\end{aligned}$$

**s.1.1.18.** Consider  $Y = \sum_{i=1}^N Z_i$ . Suppose that  $N = n$ , so that  $n$  arrivals occurred. Then we throw  $n$  independent coins with success probability  $p$ . It is clear that  $Y$  is indeed a thinned Poisson random variable.

Model the coins as a generic Bernoulli distributed random variable  $Z$ . We first need

$$\mathbb{E} \left[ e^{sZ} \right] = e^0 \mathbb{P}(Z=0) + e^s \mathbb{P}(Z=1) = (1-p) + e^s p.$$

Suppose that  $N = n$ , then since the outcomes  $Z_i$  of the coins are i.i.d.,

$$\mathbb{E} \left[ e^{s \sum_{i=1}^n Z_i} \right] = \left( \mathbb{E} \left[ e^{sZ} \right] \right)^n = (1 + p(e^s - 1))^n,$$

where we use (0.2.7d).

With (0.2.2a),

$$\begin{aligned}
\mathbb{E} \left[ e^{sY} \right] &= \mathbb{E} \left[ \sum_{n=0}^{\infty} e^{s \sum_{i=1}^n Z_i} \mathbb{1}_{N=n} \right] \\
&= \mathbb{E} \left[ \sum_{n=0}^{\infty} e^{s \sum_{i=1}^n Z_i} \mathbb{1}_{N=n} \right] \\
&= \sum_{n=0}^{\infty} \mathbb{E} \left[ e^{s \sum_{i=1}^n Z_i} \mathbb{1}_{N=n} \right] \\
&= \sum_{n=0}^{\infty} \mathbb{E} \left[ e^{s \sum_{i=1}^n Z_i} \right] \mathbb{E}[\mathbb{1}_{N=n}], \quad \text{by independence of } Z_i \text{ and } N, \\
&= \sum_{n=0}^{\infty} (1 + p(e^s - 1))^n \mathbb{P}(N=n) \\
&= \sum_{n=0}^{\infty} (1 + p(e^s - 1))^n e^{-\lambda} \frac{\lambda^n}{n!} = e^{-\lambda} \sum_{n=0}^{\infty} \frac{(1 + p(e^s - 1))^n \lambda^n}{n!} \\
&= e^{-\lambda} \exp(\lambda(1 + p(e^s - 1))) = \exp(\lambda p(e^s - 1)).
\end{aligned}$$

**s.1.1.19.** Take  $Y = \sum_{i=1}^n Z_i$  with  $Z_i \sim B(p)$ . Then,

$$M_Y(s) = \mathbb{E} \left[ e^{s \sum_{i=1}^n Z_i} \right] = \left( \mathbb{E} \left[ e^{sZ} \right] \right)^n = (1 + p(e^s - 1))^n.$$

Recall that  $p = \lambda t/n$ . Then, with (0.2.1b),

$$\lim_{n \rightarrow \infty} \left( 1 + \frac{\lambda t}{n} (e^s - 1) \right)^n = \exp(\lambda t(e^s - 1)).$$

Thus, the limit becomes equal to the moment-generating function of the Poisson distribution.

## 1.2 QUEUEING PROCESSES IN DISCRETE-TIME

We start with a description of a case to provide motivation to study queueing systems. Then we develop a set of recursions of fundamental importance to construct and simulate queueing systems. With these recursions, we analyze the efficacy of several suggestions to improve the case system. To illustrate the power of this approach, we close the section with a large number of exercises in which you develop recursions for many different queueing situations.

### *Case*

At a mental health department, five psychiatrists do intakes of future patients to determine the best treatment process for the patients. There are complaints about the time patients have to wait for their first intake; the desired waiting time is around two weeks, but the realized waiting time is sometimes more than three months. The organization considers this to be unacceptably long, but... what to do about it?

To reduce the waiting times, the five psychiatrists have various suggestions.

1. Not all psychiatrists have the same amount of time available per week to do intakes. This is not a problem during weeks when all psychiatrists are present; however, psychiatrists tend to take holidays, visit conferences, and so on. So, if the psychiatrist with the most intakes per week would go on leave, this might affect the behavior of the queue length considerably. This raises the question about the difference in the allocation of capacity allotted to the psychiatrists. What are the consequences on the distribution and average of the waiting times if they would all have the same weekly capacity?
2. The psychiatrists tend to plan their holidays consecutively, to reduce the variation in the service capacity. What if they would synchronize their holidays, to the extent possible, rather than spread their holidays?
3. Finally, suppose the psychiatrists would do 2 more intakes per week in busy times and 2 fewer in quiet weeks. Assuming that the system is stable, i.e., the average service capacity is larger than the average demand, then on average the psychiatrists would not do more intakes, i.e., their workload would not increase, but the queue length may be controlled better.

As this case is too hard to analyze by mathematical means, we need to develop a model to simulate the queueing system in discrete time. With this simulator we can evaluate the effect of these suggestions on reducing the queueing dynamics. Interestingly, the structure of the simulation is very simple, so simple that it is also an exceedingly convincing tool to communicate the results of an analysis of a queueing system to managers (and the like).

### *Recursions*

Let us start with discussing the essentials of the simulation of a queueing system. The easiest way to construct queueing processes is to 'chop up' time in periods and develop recursions for the behavior of the queue from period to period. Using fixed-sized periods has the advantage that we do not have to specify specific inter-arrival times or service times of individual customers; only the number of arrivals in a period and the number of potential services are relevant. Note

that the length of such a period depends on the context for which the model is developed. For instance, to study queueing processes at a supermarket, a period can consist of 5 minutes, while for a production environment, e.g., a job shop, it can be a day, or even a week.

Let us define:

$a_k$  = number of jobs that arrive *in* period  $k$ ,

$c_k$  = the capacity, i.e., the maximal number of jobs that can be served, during period  $k$ ,

$d_k$  = number of jobs that depart *in* period  $k$ ,

$L_k$  = number of jobs in the system at the *end* of period  $k$ .

In the sequel we also call  $a_k$  the *size of the batch* arriving in period  $k$ . Note that the definition of  $a_k$  is a bit subtle: we may assume that the arriving jobs arrive either at the start or at the end of the period. In the first case, the jobs can be served in period  $k$ , in the latter case, they *cannot* be served in period  $k$ .

Since  $L_{k-1}$  is the queue length at the *end* of period  $k-1$ , it must also be the number of customers at the *start* of period  $k$ . Assuming that jobs arriving in period  $k$  cannot be served in period  $k$ , the number of customers that depart in period  $k$  is

$$d_k = \min\{L_{k-1}, c_k\}, \quad (1.2.1a)$$

since only the jobs that are present at the start of the period, i.e.,  $L_{k-1}$ , can be served if the capacity exceeds the queue length. Now that we know the number of departures, the number at the end of period  $k$  is given by

$$L_k = L_{k-1} - d_k + a_k. \quad (1.2.1b)$$

Like this, if we are given  $L_0$ , we can obtain  $L_1$ , and from this  $L_2$ , and so on.

Note that in this type of queueing system there is not a job in service, we only count the jobs in the system at the end of a period. Thus, the number of jobs in the system and in queue coincide in this model.

**1.2.1.** Suppose that  $c_k = 7$  for all  $k$ , and  $a_1 = 5$ ,  $a_2 = 4$  and  $a_3 = 9$ ; also  $L_0 = 8$ . What are  $d_k$  and  $L_k$  for  $k \geq 1$ ?

**1.2.2.** What is the interpretation of setting  $d_k = \min\{L_{k-1} + a_k, c_k\}$  rather than definition (1.2.1a)?

**1.2.3.** Show that the scheme

$$\begin{aligned} L_k &= [L_{k-1} + a_k - c_k]^+, \\ d_k &= L_{k-1} + a_k - L_k. \end{aligned} \quad (1.2.2)$$

is equivalent to a modification of (1.2.1) in which we assume that jobs can be served in the period they arrive.

Of course we are not going to carry out these computations by hand. Typically we use company data to model the arrival process  $\{a_k\}_{k=1,2,\dots}$  and the capacity  $\{c_k\}_{k=1,2,\dots}$  and feed this data into a computer to carry out the recursions (1.2.1). If we do not have sufficient data, we make a probability model for these data and use the computer to generate random numbers with, hopefully, similar characteristics as the real data. At any rate, from this point on, we assume that it is easy, by means of computers, to obtain numbers  $a_1, \dots, a_n$  for  $n \gg 1000$ , and so on.



### Case analysis

Here we continue with the case of the five psychiatrists and analyze the proposed rules to improve the performance of the system. We mainly want to reduce the long waiting times.

As a first step in the analysis, we model the arrival process of patients as a Poisson process, cf. Section 1.1. The duration of a period is taken to be a week. The average number of arrivals per period, based on data of the company, was slightly less than 12 per week; in the simulation we set it to  $\lambda = 11.8$  per week. We model the capacity in the form of a matrix such that row  $i$  corresponds to the weekly capacity of psychiatrist  $i$ :

$$C = \begin{pmatrix} 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ 3 & 3 & 3 & \dots \\ 9 & 9 & 9 & \dots \end{pmatrix}.$$

Thus, psychiatrists 1, 2, and 3 do just one intake per week, the fourth does 3, and the fifth does 9 intakes per week. The sum over column  $k$  is the total service capacity for week  $k$  of all psychiatrists together.

With the matrix  $C$  it is simple to make other capacity schemes. A more balanced scheme would be like this:

$$C = \begin{pmatrix} 2 & 2 & 2 & \dots \\ 2 & 2 & 2 & \dots \\ 3 & 3 & 3 & \dots \\ 4 & 4 & 4 & \dots \\ 4 & 4 & 4 & \dots \end{pmatrix}.$$

Next, we include the effects of holidays on the capacity. This is easily done by setting the capacity of a certain psychiatrist to 0 in a certain week. Let us assume that just one psychiatrist is on leave in a week, each psychiatrist has one week per five weeks off, and the psychiatrists' holiday schemes rotate. To model this, we set  $C_{1,1} = C_{2,2} = \dots = C_{1,6} = C_{2,7} = \dots = 0$ , i.e.,

$$C = \begin{pmatrix} 0 & 2 & 2 & 2 & 2 & 0 & \dots \\ 2 & 0 & 2 & 2 & 2 & 2 & \dots \\ 3 & 3 & 0 & 3 & 3 & 3 & \dots \\ 4 & 4 & 4 & 0 & 4 & 4 & \dots \\ 4 & 4 & 4 & 4 & 0 & 4 & \dots \end{pmatrix}.$$

Hence, the total average capacity must be  $4/5 \cdot (2 + 2 + 3 + 4 + 4) = 12$  patients per week. The other holiday scheme—all psychiatrists take holiday in the same week—corresponds to setting entire columns to zero, i.e.,  $C_{i,5} = C_{i,10} = \dots = 0$  for week 5, 10, and so on. Note that all these variations in holiday schemes result in the same average capacity.

Now that we have modeled the arrivals and the capacities, we can use the recursions (1.2.1) to simulate the queue length process for the four different scenarios proposed by the psychiatrists, unbalanced versus balanced capacity, and spread out holidays versus simultaneous holidays.

The results are shown in Fig. 1. It is apparent that Suggestions 1 and 2 above do not significantly affect the behavior of the queue length process.

Now we consider Suggestion 3, which comes down to doing more intakes when it is busy, and fewer when it is quiet. A simple rule would be to use week's queue  $L_{n-1}$ : if  $L_{n-1} < 12$ , serve  $e$

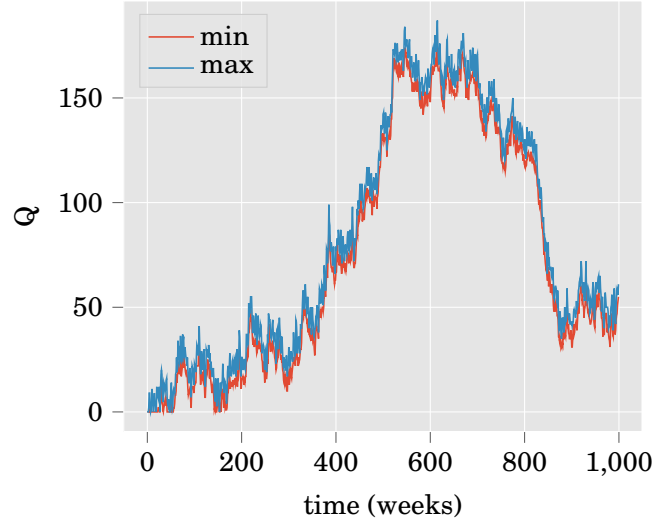


Figure 1: Effect of capacity and holiday plans. We plot for each time point the maximum and the minimum queue length for each of the policies. Apparently, the effect of each of these policies is, for all practical purposes, negligible.

intakes less (in other words, when the service capacity of week  $k$  is larger than the queue length serve  $e$  less), when  $L_{n-1} > 24$ , serve  $e$  more. Here,  $e = 1$  or  $2$ , or perhaps a larger number. The larger  $e$ , the larger the control we exercise.

Let's consider three different control levels,  $e = 1$ ,  $e = 2$ , and  $e = 5$ ; thus in the last case all psychiatrists do five extra intakes. The previous simulation shows that it is safe to disregard the holiday plans, so just assume a flat service capacity of 12 intakes a week. The results, see Fig. 2, show a striking difference indeed. The queue does not explode anymore, and already taking  $e = 1$  has a large influence.

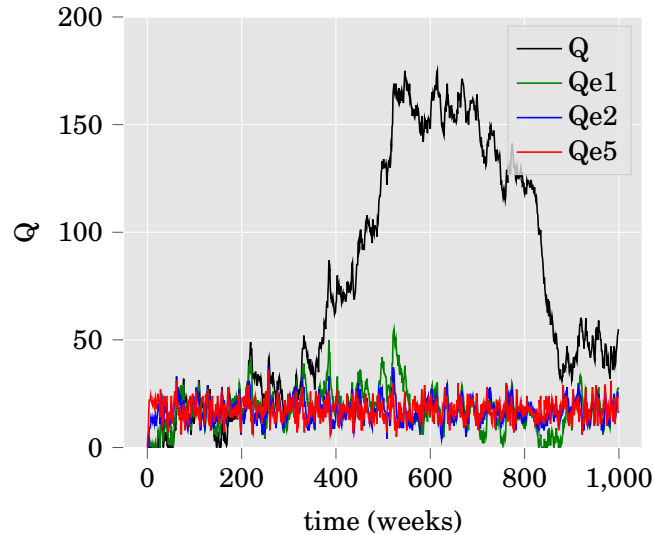


Figure 2: Controlling the number of intakes. Clearly, adapting the service rate ‘does wonders’ to control the queue length.

From this simulation experiment, we learn that changing holiday plans or spreading the work over multiple servers, i.e., psychiatrists, does not significantly affect the queueing behavior.

However, controlling the service rate as a function of the queue length improves the situation quite dramatically.

### Conclusion

From the above case we learn that even with these (deceitfully) simple recursions, we can obtain considerable insight into the behavior and performance of this quite complicated controlled queueing system<sup>1</sup>. In general, the study of queueing systems is focused on studying the probabilistic properties of the queueing length process and related concepts such as waiting time, server occupancy, fraction of customers lost, and so on. Once we have constructed the queueing process, we can compute all relevant performance measures, such as the average waiting time. If it turns out that the performance of the system is not according to what we desire, we can change parts of the system with the aim to improve the situation and assess the effect of this change. For instance, if the average waiting time is too long, we might want to add service capacity. With simulation it is easy to study, hence evaluate, the effect of such decisions.

### Exercises

The reader should understand from the above case that, once we have the recursions, we can analyze the system and make, for instance, plots to evaluate suggestions for improvement. Thus, formulating the recursions is crucial to construct queueing processes. For this reason, many of the exercises below concentrate on obtaining recursions for specific queueing systems.

It may be that the recursions you find are not identical to the recursions in the solution. The reason is that the assumptions you make might not be equal to the ones I make. I don't quite know how to get out of this paradoxical situation. In a sense, to completely specify the model, we need the recursions. However, if the problem statement would contain the recursions, there would be nothing left to practice anymore. Another way is to make the problem description five times as long, but this is also undesirable. So, let's be pragmatic: the aim is that you practice with modeling, and that you learn from the solutions. If you obtain *reasonable* recursions, but they are different from mine, then your answer is just as good.

**1.2.4 (Queue with Blocking).** *A queueing system under daily review, i.e., at the end of the day the queue length is measured. We assume that at the end of the day no jobs are still in service. We assume that jobs that arrive at day  $k$  cannot be served in day  $k$ . The queue length cannot exceed level  $K$ . Formulate a set of recursions to cover this case. What is the loss per period? What is the fraction of jobs lost?*

**1.2.5 (Estimating the lead time distribution).** *Take  $d_k = \min\{L_{k-1} + a_k, c_k\}$ , and assume that jobs are served in FIFO sequence. Find an expression for the shortest possible waiting time  $W_{-,k}$  of a job that arrives at time  $k$ , and an expression for the largest possible waiting time  $W_{+,k}$ .*

**1.2.6 (Yield loss).** *A machine produces items to serve customer demand. A fraction  $p$  of the items produced in a period turns out to be faulty, and has to be made anew. Develop a set of recursions to model the queue of customers waiting to be served from the machine.*

<sup>1</sup> If you doubt the value of simulation, try to develop a mathematical method to analyze multi-server queueing systems with vacations, of which this case is an example.

**1.2.7 (Rework).** *A machine produces items, but a fraction  $p$  of the items does not meet the quality requirements after the first pass at the server, but it requires a second pass. Assume that the repair of a faulty item requires half of the work of a new job, and that the faulty jobs are processed with priority over the new jobs. Also assume that faulty items do not need more than one repair (hence, faulty items that are repaired cannot be faulty anymore). Develop a set of recursions to model this case.*

*We remark that there are of course many different policies to treat rework. One possibility is to collect faulty items until there are  $N$  items, and then repair the entire batch of  $N$  items.*

**1.2.8 (Cost models).** *A single-server queueing station processes customers. At the start of a period the server capacity is chosen, so that for period  $k$  the capacity is  $c_k$ . Demand that arrives in a period can be served in that period. It costs  $\beta$  per unit time per unit processing capacity to operate the machine, i.e., to have it switched on. There is also a cost  $h$  per unit time per job in the system. Make a cost model to analyze the long-run average cost for this case.*

**1.2.9 (N-policies).** *A machine can switch on and off. If the queue length hits  $N$ , the machine switches on, and if the system becomes empty, the machine switches off. It costs  $K$  to switch on the machine. There is also a cost  $\beta$  per unit time while the machine is switched on, and it costs  $h$  per unit time per customer in the system. Make a cost model.*

**1.2.10 (Queue length dependent server capacity).** *How would you model (in terms of recursions) a server whose capacity depends on the queue length? Consider, as an example, a rule such that the server only works if the queue is larger than a threshold  $t$ .*

**1.2.11 (Fair queueing).** *One server serves two queues. Each queue receives service capacity in proportion to the queue length. Derive a set of recursions to analyze this situation.*

**1.2.12 (Priority queueing).** *An interesting situation is a system with two queues served by one server, but such that one queue, queue A, gets priority over the other queue. Again find a set of recursions to describe this case.*

**1.2.13 (Queues with reserved service capacity).** *Consider a single-server that serves two parallel queues A and B. Each queue has a minimal service capacity every period,  $r_A$  for queue A and  $r_B$  for queue B. Reserved capacity unused for one queue can be used to serve the other queue. Any extra capacity beyond the reserved capacity is given to queue A with priority. Formulate a set of recursions to analyze this situation.*

**1.2.14 (Queue with protected service capacity and lost capacity).** *Consider a single-server that serves two parallel queues A and B. Each queue receives a minimal service capacity every period. Reserved capacity unused for one queue cannot be used to serve the other queue. Any extra capacity beyond the reserved capacity is given to queue A with priority. Formulate a set of recursions to analyze this situation.*

*Let  $r_A$  be the reserved capacity for queue A, and likewise for  $r_B$ . We assume of course that  $c_k \geq r_A + r_B$ , for all  $k$ .*

**1.2.15 (Tandem networks).** *Consider a production network with two production stations in tandem, that is, the jobs processed by station A are in the next period to the downstream Station B. Extend the recursions of (1.2.1) to simulate this situation.*

**1.2.16 (A tandem queue with blocking).** *Consider a production network with two production stations in tandem with blocking: when the intermediate queue, i.e., the queue in front of Station*

$B$ , exceeds some level  $M$ , then station  $A$  has to stop producing, and when  $L_k^B < M$  station  $A$  is not allowed to produce more than the intermediate queue can contain. Extend the recursions of (1.2.1) to simulate this situation.

**1.2.17** (Merging departure streams). Consider another production situation with two machines,  $A$  and  $B$  say, that send their products to Station  $C$ . Derive a set of recursion relations to simulate this system.

**1.2.18** (Merging incoming streams). Consider a single-server queue that serves two customer 'streams' in a FIFO discipline. Thus, both streams enter one queue that is served by the server. Let  $\{a_k^a\}$  be the number of arrivals of stream  $a$  in period  $k$  and  $\{a_k^b\}$  be the number of arrivals of stream  $b$ . Find a set of recursions by which it becomes possible to analyze the waiting time distribution of each of the streams. Assume that the service capacity  $c$  is constant for all periods, and that jobs that arrive in period  $k$  can also be served in period  $k$ .

**1.2.19** (Splitting streams). Consider a machine (a paint mixing machine) that produces products for two separate downstream machines  $A$  and  $B$  (two different paint packaging machines), each with its own queue. Suppose we want to analyze the queue in front of station  $A$ . For this we need to know the arrivals to station  $A$ , that is, the departures of the mixing station that go to station  $A$ . Provide a set of recursions to simulate this system.

**1.2.20** (Queue with setups, not obligatory). One server serves two parallel queues, one at a time. After serving one queue until it is empty, the server moves to the other queue. The change of queue requires one period setup time.

### Hints

**h.1.2.5.** Consider a numerical example. Suppose  $L_{k-1} = 20$ . Suppose that the capacity is  $c_k = 3$  for all  $k$ . Then a job that arrives in the  $k$ th period, must wait at least  $20/3$  (plus rounding) periods before it can leave the system. Now generalize this numerical example.

### Solutions

**s.1.2.1.**  $d_1 = 7$ ,  $L_1 = 8 - 7 + 5 = 6$ ,  $d_2 = 6$ ,  $L_2 = 6 - 6 + 4 = 4$ ,  $d_3 = 4$ ,  $L_3 = 4 - 4 + 9 = 9$ , and so on.

**s.1.2.2.** The assumption is that the jobs arrive at the start of period  $k$ , before service in period  $k$  starts, rather than at the end of the period. Therefore the arrivals at period  $k$  can also be served during period  $k$ .

**s.1.2.3.** Starting from ,

$$\begin{aligned} d_k &= \min\{L_{k-1} + a_k, c_k\}, \\ L_k &= L_{k-1} + a_k - d_k, \end{aligned}$$

we get

$$\begin{aligned} L_k &= L_{k-1} + a_k - d_k \\ &= L_{k-1} + a_k - \min\{L_{k-1} + a_k, c_k\} \\ &= \max\{L_{k-1} + a_k - c_k, 0\}. \end{aligned}$$

In code it looks like this.

```

>>> a = [0, 10, 3, 6]
>>> c = [0, 5, 5, 5]
>>> L = [0] * len(a)
>>> d = [0] * len(a)

>>> for k in range(1, len(a)):
...     L[k] = max(L[k - 1] + a[k] - c[k], 0)
...     d[k] = L[k - 1] + a[k] - L[k]
...

>>> print(L)
[0, 5, 3, 4]
>>> print(d)
[0, 5, 5, 5]

```

**s.1.2.4.** All jobs that arrive such that the queue becomes larger than  $K$  must be dropped.

First  $d_k = \min\{L_{k-1}, c_k\}$ . Then,  $L'_k = L_{k-1} + a_k - d_k$  is the queue without blocking. Then  $L_k = \min\{L'_k, K\}$  is the queue with blocking. Finally, the loss  $l_k = L'_k - L_k$ , i.e., the excess arrivals. The fraction lost is  $l_k/a_k$ .

```

>>> a = [0, 10, 3, 6]
>>> c = [0, 5, 5, 5]
>>> L = [0] * len(a)
>>> d = [0] * len(a)
>>> l = [0] * len(a) # loss

>>> K = 8

>>> for k in range(1, len(a)):
...     d[k] = min(L[k - 1], c[k])
...     Lp = L[k - 1] + a[k] - d[k]
...     L[k] = min(Lp, K)
...     l[k] = Lp - L[k]
...

>>> print(L)
[0, 8, 6, 7]
>>> print(l)
[0, 2, 0, 0]

```

**s.1.2.5.** Let's tag the first customer that arrives in period  $k$ . This tagged customer sees  $L_{k-1}$  customers in the system, hence the tagged customer's service can only start after all  $L_{k-1}$  customers have been served (assuming FIFO scheduling of course). Now, if  $L_{k-1} - c_k \geq 0$ , there are still people in front of the tagged customer, either in queue or in service. In fact, as long as  $c_k + c_{k+1} + \dots + c_m \leq L_{k-1}$ , there are still customers in front of the tagged customer. Therefore, when the tagged customer leaves the system, the inequality  $c_k + c_{k+1} + \dots + c_m > L_{k-1}$  must hold. (To check your understanding, explain that  $c_k + c_{k+1} + \dots + c_m \geq L_{k-1}$  is not correct.)

We can also tag the last customer that arrives in period  $k$ . This customer will certainly have left if  $m$  is such that  $c_k + \dots + c_m \geq L_{k-1} + a_k$ .

In formulas, the above comes down to the following. A job that arrives in period  $k$  cannot be served before period

$$W_{-,k} := \min \left\{ m : \sum_{i=k}^{k+m} c_i > L_{k-1} \right\},$$

and it must have been served before period

$$W_{+,k} := \min \left\{ m : \sum_{i=k}^{k+m} c_i \geq L_{k-1} + a_k \right\}.$$

Thus, the waiting time of jobs arriving in period  $k$  must lie in the interval  $[W_{-,k}, W_{+,k}]$ .

Here is some code to see how it works. It is less easy then you might think to get the details correct.

```
>>> a = [3] * 10
>>> c = [7] * len(a)
>>> L = [0] * len(a)
>>> d = [0] * len(a)

>>> L[0] = 20

>>> for k in range(1, len(a)):
...     d[k] = min(L[k - 1] + a[k], c[k])
...     L[k] = L[k - 1] + a[k] - d[k]
...

>>> print(L)
[20, 16, 12, 8, 4, 0, 0, 0, 0, 0]

>>> k = 1
>>> m = 0
>>> tot_service = c[k + m]
>>> print(m, L[k - 1] - tot_service)
0 13
>>> while tot_service <= L[k - 1]:
...     m += 1
...     tot_service += c[k + m]
...     print(m, L[k - 1] - tot_service)
...
1 6
2 -1
>>> W_min = m
>>> print(W_min)
2
>>> print(L[k - 1] / c[0])
2.857142857142857
```

```

>>> print(m, L[k - 1] + a[k] - tot_service)
2 2
>>> while tot_service <= L[k - 1] + a[k]:
...     m += 1
...     tot_service += c[k + m]
...     print(m, L[k - 1] + a[k] - tot_service)
...
3 -5
>>> W_max = m
>>> print(W_max)
3
>>> print((L[k - 1] + a[k]) / c[0])
3.2857142857142856

```

**s.1.2.6.** The machine amount produced in period  $k$  is  $d_k$ . Thus,  $pd_k$  is the amount lost, neglecting rounding errors for the moment. Thus,  $(1-p)d_k$  items can be used to serve customer demand. Therefore

$$d_k = \min\{L_{k-1}, c_k\},$$

$$L_k = L_{k-1} - d_k(1-p) + a'_k.$$

Can you use these recursions to show that the long-run average service capacity  $n^{-1} \sum_{i=1}^n c_i$  must be larger than  $\lambda(1+p)$ ?

If you like, you can incorporate time-dependent failure rates  $\{p_k\}$  too. Whether this makes practical sense depends on the context of course.

```

>>> a = [0, 4, 8, 2, 1]
>>> c = [3] * len(a)
>>> L = [0] * len(a)
>>> d = [0] * len(a)
>>> p = 0.2

>>> L[0] = 2

>>> for k in range(1, len(a)):
...     d[k] = (1 - p) * min(L[k - 1], c[k])
...     L[k] = L[k - 1] + a[k] - d[k]
...
>>> print(L)
[2, 4.4, 10.0, 9.6, 8.2]

```

**s.1.2.7.** Suppose again that a fraction  $p$  is faulty. Since these faulty items require less processing time than a new job, the service capacity  $c_k$ , i.e., the number of jobs that can be processed in period  $k$ , is a bit bigger; part of the capacity is spent on new jobs, but another part is spent on the faulty jobs. By the assumptions above, the repair of a faulty requires half of the work of a new job, and the faulty jobs are processed with priority over the new jobs. Assume queue  $A$  contains the faulty items, and queue  $B$  the new jobs. Then the recursions become:

$$d_{A,k} = \min\{L_{A,k-1}, 2c_k\}, \quad (\text{as faulty jobs require half of the processing time})$$



$$\begin{aligned}
c_{B,k} &= c_k - d_{A,k}/2, \\
d_{B,k} &= \min\{L_{B,k-1}, c_{B,k}\}, \\
a_{A,k} &= p d_{B,k-1}, \\
a_{B,k} &= (1-p) d_{B,k-1}, \\
L_{A,k} &= L_{A,k-1} + a_{A,k} - d_{A,k}, \\
L_{B,k} &= L_{B,k-1} + a_{B,k} - d_{B,k}.
\end{aligned}$$

In code:

```

>>> a = [0, 4, 8, 2, 1]
>>> c = [3] * len(a)
>>> L_A = [0] * len(a)
>>> d_A = [0] * len(a)
>>> L_B = [0] * len(a)
>>> d_B = [0] * len(a)
>>> p = 0.2

>>> L_A[0] = 2
>>> L_B[0] = 8

>>> for k in range(1, len(a)):
...     d_A[k] = min(L_A[k - 1], 2 * c[k])
...     c_B = c[k] - d_A[k] / 2
...     d_B[k] = min(L_B[k - 1], c_B)
...     a_A = p * d_B[k - 1]
...     a_B = a[k]
...     L_A[k] = L_A[k - 1] + a_A - d_A[k]
...     L_B[k] = L_B[k - 1] + a_B - d_B[k]
...
>>> print(L_A)
[2, 0.0, 0.4, 0.6, 0.5599999999999999]
>>> print(L_B)
[8, 10.0, 15.0, 14.2, 12.5]

```

**s.1.2.8.** First consider the dynamics of the queue. Since the capacity is chosen at the start of the period (the machine is switched on for  $c_k$  units even if there is less demand):

$$\begin{aligned}
d_k &= \min\{L_{k-1} + a_k, c_k\}, \\
L_k &= L_{k-1} + a_k - d_k.
\end{aligned}$$

The cost to operate the server during period  $k$  is  $\beta c_k$ . Thus, the total cost up to some time  $T$  for the server must be  $\beta \sum_{k=1}^T c_k$ . In period  $k$  we also have to pay  $h L_k$ , since  $h$  is the cost per customer per period in the system. Thus, the long-run average cost is

$$\frac{1}{T} \sum_{k=1}^T (\beta c_k + h L_k).$$

It is an interesting problem to find a policy that minimizes (the expectation of) this cost. The policy is such that the capacity for period  $k$  can be chosen based on the queue length  $L_{k-1}$  and *estimates* of the demands  $\hat{d}_k, \hat{d}_{k+1}, \dots$ . This problem is not easy, as far as I can see.

**s.1.2.9.** First we need to implement the N-policy. For this we need an extra variable to keep track of the state of the server. Let  $I_k = 1$  if the machine is on in period  $k$  and  $I_k = 0$  if it is off. Then  $\{I_k\}$  must satisfy the relation

$$I_{k+1} = \begin{cases} 1 & \text{if } L_k \geq N, \\ I_k & \text{if } 0 < L_k < N, \\ 0 & \text{if } L_k = 0, \end{cases}$$

and assume that  $I_0 = 0$  at the start, i.e., the machine is off. Thus, we can write:

$$I_{k+1} = \mathbb{1}_{L_k \geq N} + I_k \mathbb{1}_{0 < L_k < N} + 0 \cdot \mathbb{1}_{L_k = 0}.$$

With  $I_k$  it follows that  $d_k = \min\{L_{k-1}, I_k c_k\}$ , from which  $L_k$  follows, and so on.

The machine cost for period  $k$  is  $\beta I_k$ , because only when the machine is on we have to pay  $\beta$ , and the queueing cost is  $hL_k$ . To determine the total switching cost is harder as we need to determine how often the machine has been switched on up to time  $T$ . Observe that the machine is switched on in period  $k$  if  $I_{k-1} = 0$  and  $I_k = 1$ . Thus, whenever  $I_k - I_{k-1} = 1$  the machine is switched on, when  $I_k - I_{k-1} = 0$  the state of the machine remains the same, and if  $I_k - I_{k-1} = -1$  the machine is switched off. In other words  $\max\{I_k - I_{k-1}, 0\}$  captures what we need. The total cost up to time  $T$  becomes:

$$\sum_{k=1}^T (\beta I_k + hL_k + K \max\{I_k - I_{k-1}, 0\}).$$

**s.1.2.10.** One model could be to let the server only switch on when the queue is larger than some threshold  $t$ , and when the server is on, it works at rate  $c$  per period. In that case,  $c_k = c \mathbb{1}_{L_{k-1} > t}$ .

**s.1.2.11.** Let  $c_k^i$  be the capacity allocated to queue  $i$  in period  $k$ . The fair rule gives that

$$c_k^1 = \frac{L_{k-1}^1}{L_{k-1}^1 + L_{k-1}^2} c = c - c_k^2.$$

Then,

$$\begin{aligned} d_k^1 &= \min\{L_{k-1}^1, c_k^1\}, \\ L_k^1 &= L_{k-1}^1 + a_k^1 - d_k^1, \end{aligned}$$

and likewise for the other queue.

**s.1.2.12.** The rules below implement a strict priority rule for jobs of type A, i.e., jobs sent into queue A.

$$\begin{aligned} d_{A,k} &= \min\{L_{A,k-1}, c_k\}, \\ c_{B,k} &= c_k - d_{A,k}, \\ d_{B,k} &= \min\{L_{B,k-1}, c_{B,k}\}, \\ L_{A,k} &= L_{A,k-1} + a_{A,k} - d_{A,k}, \\ L_{B,k} &= L_{B,k-1} + a_{B,k} - d_{B,k}. \end{aligned}$$

As an aside, another interesting rule to distribute the capacity  $c_k$  over the queues could be based on the principle of *equal division of the contested sum*. This principle is based on game-theoretic ideas. Aumann and Maschler applied this principle to clarify certain division rules discussed in the Talmud to divide the legacy among a number of inheritors, each having a different claim size.

**s.1.2.13.** First determine how much capacity queue  $B$  minimally needs in period  $k$ :

$$c_{B,k} = \min\{L_{B,k-1}, r_B\}.$$

Observe that, since  $c_k \geq r_A + r_B$ , this rule ensures that queue A receives at least its reserved capacity  $r_A$ .

Since queue A is served with priority, we first give all capacity, except what queue B minimally needs, to queue A:

$$d_{A,k} = \min\{L_{A,k-1}, c_k - c_{B,k}\}.$$

And then we can give any leftover capacity to queue B, if needed.

$$d_{B,k} = \min\{L_{B,k-1}, c_k - d_{A,k}\}.$$

An example is the weekly capacity offered by a psychiatrist at a hospital. Part of the weekly capacity is reserved/allocated/assigned to serve certain patient groups. For instance, each week the psychiatrist does at most five intakes of new patients, provided there are any, and the rest of the capacity is used to treat other patients. The existing patients can also be divided in different groups, each receiving a minimal capacity. If there are fewer patients of some group, then the capacity can be planned/given to other patient groups.

**s.1.2.14.** Queue A can use all capacity, except what is reserved for queue B:

$$d_{A,k} = \min\{L_{A,k-1}, c_k - r_B\}.$$

Observe that, since  $c_k \geq r_A + r_B$ , this rule ensures that queue A receives at least its reserved capacity  $r_A$ .

Queue B cannot receive more than  $c_k - r_A$ , since  $r_A$  is allocated to queue A, and if queue A does not use all of  $r_A$ , then the surplus is lost. Also, queue B cannot get more than  $c_k - d_{A,k}$  as this is what remains after serving queue A. Thus, letting  $c_{B,k} = \min\{c_k - r_A, c_k - d_{A,k}\} = c_k - \max\{r_A, d_{A,k}\}$ , we see that for queue B:

$$d_{B,k} = \min\{L_{B,k-1}, c_{B,k}\}.$$

An example can be the operation room of a hospital. There is a weekly capacity, part of the capacity is reserved for emergencies. It might not be possible to assign this reserved capacity to other patient groups, because it should be available at all times for emergency patients. A result of this is that unused capacity is lost.

In practice it may not be as extreme as in the model, but still part of the unused capacity is lost. ‘Use it, or lose it’, is what often, but not always, applies to service capacity.

**s.1.2.15.** Let  $a_k$  be the external arrivals at station A. Then:

$$\begin{aligned} d_k^A &= \min\{L_{k-1}^A, c_k^A\}, \\ L_k^A &= L_{k-1}^A - d_k^A + a_k. \end{aligned} \tag{1.2.3}$$

The departures of the first station during period  $k$  are the arrivals at station B at the end of period  $k$ , i.e.,  $a_k^B = d_k^A$ . Thus,

$$\begin{aligned} a_k^B &= d_k^A, \\ d_k^B &= \min\{L_{k-1}^B, c_k^B\}, \\ L_k^B &= L_{k-1}^B - d_k^B + a_k^B. \end{aligned} \tag{1.2.4}$$

**s.1.2.16.**

$$\begin{aligned}
d_k^A &= \min\{L_{k-1}^A, c_k^A, M - L_{k-1}^B\}, \\
L_k^A &= L_{k-1}^A - d_k^A + a_k, \\
a_k^B &= d_k^A, \quad \text{ensures all jobs first pass station A and then station B} \\
d_k^B &= \min\{L_{k-1}^B, c_k^B\}, \\
L_k^B &= L_{k-1}^B - d_k^B + a_k^B.
\end{aligned} \tag{1.2.5}$$

This is a bit subtle: since there is room  $M - L_{k-1}^B$  at the intermediate buffer and  $d_k^A \leq M - L_{k-1}^B$ , we know that in the worst case, i.e., when  $c_k^B = 0$ , still  $L_k^B = L_{k-1}^B + d_k^A$ . Thus, we are sure that the queue length of the intermediate queue will not exceed  $M$ .

There is still a small problem: What if for the first initial periods  $M < L_{k-1}^B$ . Then  $M - L_{k-1}^B < 0$  and then by the specification above,  $d_k^A < 0$ . This is not what we want. Therefore,

$$d_k^A = \min\{L_{k-1}^A, c_k^A, \max\{M - L_{k-1}^B, 0\}\}.$$

**s.1.2.17.** Realize that Stations A and B have their own arrivals.

$$\begin{aligned}
d_k^A &= \min\{L_{k-1}^A, c_k^A\}, \\
L_k^A &= L_{k-1}^A - d_k^A + a_k^A, \\
d_k^B &= \min\{L_{k-1}^B, c_k^B\}, \\
L_k^B &= L_{k-1}^B - d_k^B + a_k^B, \\
a_k^C &= d_k^A + d_k^B, \\
d_k^C &= \min\{L_{k-1}^C, c_k^C\}, \\
L_k^C &= L_{k-1}^C - d_k^C + a_k^C.
\end{aligned} \tag{1.2.6}$$

**s.1.2.18.** The behavior of the queue length process is easy:

$$\begin{aligned}
d_k &= \min\{L_{k-1} + a_k^a + a_k^b, c\}, \\
L_k &= L_{k-1} + a_k^a + a_k^b - d_k.
\end{aligned}$$

To determine the waiting times, observe that any arrival in period  $k$ , independent of the stream, has to wait until all jobs at the start of the period in queue, i.e.,  $L_{k-1} - d_k$ , are cleared; note that we assume here that the jobs served in period  $k$  depart at the start of the interval. Thus, the minimal waiting time is  $W_{k,-} = \lfloor L_{k-1}/c \rfloor$ . Similarly, the maximal waiting time is  $W_{k,+} = \lfloor (L_{k-1} + a_k^a + a_k^b)/c \rfloor$ .

The remaining problem is to make a model to ‘distribute’ the time between  $W_{k,-}$  and  $W_{k,+}$  over the two streams.

A simple model is to assume that the waiting time is averaged over the jobs. Then each job perceives a waiting time of

$$\frac{W_{k,-} + W_{k,+}}{2}.$$

Another way is to give priority to  $a$  customers, but only for the jobs that arrive in this period. (Hence, this is different from the priority queue. In that case priority customers can overtake customers that arrived earlier. In the present case this is not allowed, jobs of type  $a$  that arrive in period  $k$  cannot overtake  $b$  jobs that arrived prior to period  $k$ .) Making this completely explicit

(so that the recursion can be fed to the computer) requires a bit of work however. It is important to understand the trick we will discuss now because we will use it to model queueing systems with batching. Observe that the first job of the  $a$  stream only has to wait for  $W_{k,-}$ , the second job must wait for  $W_{k,-} + 1$ , and so on. Thus, the waiting time  $W_k^a$  for the  $a_k^a$  items is such that

$$W_{k,-}^a := \lfloor L_{k-1}/c \rfloor \leq W_k^a \leq \lfloor (L_{k-1} + a_k^a)/c \rfloor =: W_{k,+}^a.$$

Similarly, for the  $b$  jobs must be

$$W_{k,-}^b := \lfloor (L_{k-1} + a_k^a)/c \rfloor \leq W_k^b \leq \lfloor (L_{k-1} + a_k^a + a_k^b)/c \rfloor =: W_{k,+}^b.$$

Note that  $W_{k,+}^a = W_{k,-}^b$ . It is then sensible to set

$$W_k^a = \frac{W_{k,-}^a + W_{k,+}^a}{2},$$

$$W_k^b = \frac{W_{k,-}^b + W_{k,+}^b}{2}.$$

**s.1.2.19.** 1. Realize that the recursions of (1.2.1), applied to the queueing situation at the first machine, provide us with the total number of departures  $d_k$  during period  $k$ . However, it does not tell us about the type of these departures. Thus, to compute the queue in front of station A, we need to know the number of departures of type A, rather than the total number of departures of the first station.

2. It is reasonable that the number of jobs of type A in queue at the first station is equal to

$$L_k \frac{\lambda_A}{\lambda_A + \lambda_B}.$$

It is therefore reasonable to assume that the capacity  $c_k$  of the first station is also shared in this proportion to type A and B jobs. Thus, the number of departures to station A is

$$d_k(A) = \frac{\lambda_A}{\lambda_A + \lambda_B} \min\{L_{k-1}, c_k\}.$$

The rest of the recursions is very similar to what we did in earlier exercises.

**s.1.2.20.** We need an extra variable  $p_k$  that specifies which queue is being served. If  $p_k = 0$ , the server is moving from one queue to the other, if  $p_k = 1$ , the server is at queue 1, and if  $p_k = 2$  it is at queue 2. Then, with

$$d_k^1 = \min\{L_{k-1}^1, c_k^1 \mathbb{1}_{p_k=1}\},$$

$$d_k^2 = \min\{L_{k-1}^2, c_k^2 \mathbb{1}_{p_k=2}\},$$

we can specify the evolution of the queue length processes. So it remains to deal with  $p$ . For this, we use a ‘truth table’ to compute  $p_{k+1}$  as a function of  $p_k$  and whether  $L_k^1 > 0$  or not and  $L_k^2 > 0$  or not.

$\mathbb{1}_{L_k^1 > 0}$	$\mathbb{1}_{L_k^2 > 0}$	$p_k$	$p_{k+1}$
0	0	0	0
0	0	1	1
0	0	2	2
1	0	0	1
1	0	1	1
1	0	2	0 (switch over time)
0	1	0	2
0	1	1	0 (switch over time)
0	1	2	2
1	1	0	1 (to break ties)
1	1	1	1
1	1	2	2

### 1.3 EXPONENTIAL DISTRIBUTION

In Section 1.1 we introduced the Poisson process as a natural model of the (random) number of jobs arriving during intervals of time. As we will see in the sections to come, we can model a single-server queueing system in continuous time if we specify the (probability) distribution of the inter-arrival times, i.e., the time between consecutive arrival epochs of jobs. A particularly fruitful model for the distribution of the inter-arrival times is the exponential distribution because, as it turns out, it is intimately related to the Poisson distribution. Besides explaining this relation, we derive many useful properties of the exponential distribution, in particular, that it is *memoryless*.

#### Theory and Exercises

We say that  $X$  is an *exponentially distributed* random variable with mean  $1/\lambda$  if

$$P(X \leq t) = 1 - e^{-\lambda t},$$

and then we write  $X \sim \text{Exp}(\lambda)$ .

The Poisson process  $N$  and exponentially distributed inter-arrival times are intimately related: A counting process  $\{N(t)\}$  is a *Poisson process* with rate  $\lambda$  if and only if the inter-arrival times  $\{X_i\}$  are i.i.d. with common random variable  $X \sim \text{Exp}(\lambda)$ , in short,  $X_i \sim \text{Exp}(\lambda) \Leftrightarrow N(t) \sim P(\lambda t)$ . We next provide further relations between the Poisson distribution and the exponential distribution.

**1.3.1.** In the above expression for  $P(X \leq t)$  we require that  $\lambda > 0$ . What would happen if you would allow  $\lambda$  to be zero or negative?

**1.3.2.** If the random variable  $X \sim \text{Exp}(\lambda)$ , show that its mean  $E[X] = \frac{1}{\lambda}$ .

**1.3.3.** If  $X \sim \text{Exp}(\lambda)$ , show that its second moment  $E[X^2] = \frac{2}{\lambda^2}$ .

**1.3.4.** If  $X \sim \text{Exp}(\lambda)$ , show that the variance  $V[X] = \lambda^{-2}$ .

**1.3.5.** Suppose  $X \sim \text{Exp}(\lambda)$ . Show that, only when  $t < \lambda$ , the moment-generating function  $M_X(t)$  is finite.

**1.3.6.** If  $X \sim \text{Exp}(\lambda)$ , show that its moment-generating function is equal to

$$M_X(t) = \frac{\lambda}{\lambda - t}.$$

**1.3.7.** Use the moment-generating function of  $X \sim \text{Exp}(\lambda)$  to show that

$$E[X] = \frac{1}{\lambda}, \quad E[X^2] = \frac{2}{\lambda^2}.$$

**1.3.8.** Show that, when  $X \sim \text{Exp}(\lambda)$ , the SCV of  $X$  is 1.

We now provide a number of relations between the Poisson distribution and the exponential distribution to conclude that a process  $N_\lambda$  is a Poisson process with rate  $\lambda$  iff the inter-arrival times  $\{X_i\}$  between individual jobs are i.i.d.  $\sim \text{Exp}(\lambda)$ .

**1.3.9.** If  $N_\lambda$  is a Poisson process with rate  $\lambda$ , show that the time  $X_1$  to the first arriving job is  $\text{Exp}(\lambda)$ .

**1.3.10.** Assume that inter-arrival times  $\{X_i\}$  are i.i.d. and  $\sim \text{Exp}(\lambda)$ . Let the arrival time of the  $i$ th job be  $A_i = \sum_{k=1}^i X_k$ . Show that

$$E[A_i] = \frac{i}{\lambda}.$$

**1.3.11.** Let  $A_i$  be the arrival time of customer  $i$  and set  $A_0 = 0$ . Assume that the inter-arrival times  $\{X_i\}$  are i.i.d. with exponential distribution with mean  $1/\lambda$  for some  $\lambda > 0$ . Prove that  $A_i$  has density

$$f_{A_i}(t) = \lambda e^{-\lambda t} \frac{(\lambda t)^{i-1}}{(i-1)!}.$$

**1.3.12.** Use the density  $f_{A_i}$  of [1.3.11] to show that  $E[A_i] = i/\lambda$ .

**1.3.13.** If the inter-arrival times  $\{X_i\}$  are i.i.d.  $\sim \text{Exp}(\lambda)$ , prove that the number  $N(t)$  of arrivals during the interval  $[0, t]$  is Poisson distributed.

We now introduce another fundamental concept. A random variable  $X$  is called *memoryless* when it satisfies

$$P(X > t + h | X > t) = P(X > h).$$

In words, the probability that  $X$  is larger than some time  $t + h$ , conditional on it being larger than a time  $t$ , is equal to the probability that  $X$  is larger than  $h$ .

**1.3.14.** Show that  $X \sim \text{Exp}(\lambda)$  is memoryless.

In fact, it can be shown that only exponential random variables have the memoryless property. The proof of this fact requires quite some work; we refer the reader to the literature if s/he wants to check this, see e.g. [Yushkevich and Dynkin \[1969, Appendix 3\]](#).

**1.3.15.** If  $X \sim \text{Exp}(\lambda)$  and  $S \sim \text{Exp}(\mu)$ , and  $X$  and  $S$  are independent, show that

$$Z = \min\{X, S\} \sim \text{Exp}(\lambda + \mu),$$

hence  $E[Z] = (\lambda + \mu)^{-1}$ .

**1.3.16.** If  $X \sim \text{Exp}(\lambda)$ ,  $S \sim \text{Exp}(\mu)$ , and  $X$  and  $S$  are independent, show that

$$P(X \leq S) = \frac{\lambda}{\lambda + \mu}.$$

*Hints*

**h.1.3.1.** The interpretation of the function  $1 - e^{-\lambda t}$  is a probability. What are the consequences of this? What happens if  $\lambda = 0$ ?

**h.1.3.2.**

$$E[X] = \int_0^\infty t f(t) dt = \int_0^\infty t \lambda e^{-\lambda t} dt,$$

where  $f(t) = \lambda e^{-\lambda t}$  is the density function of the distribution function  $F$  of  $X$ . Now solve the integral.

**h.1.3.3.**

$$E[X^2] = \int_0^\infty t^2 \lambda e^{-\lambda t} dt = \frac{2}{\lambda^2}.$$

**h.1.3.4.** Use (0.2.4).

**h.1.3.6.**

$$M_X(t) = E[\exp(tX)] = \int_0^\infty e^{tx} f(x) dx.$$

**h.1.3.7.** Use (0.2.7b) and (0.2.7c).

**h.1.3.8.** Use (1.1.4)

**h.1.3.9.** What is the meaning of  $P(N(t) = 0)$ ?

**h.1.3.10.** What is  $E[X + Y]$  for some general random variables (each with finite mean)?

**h.1.3.11.** Check the result for  $i = 1$  by filling in  $i = 1$  (just to be sure that you have read the formula right), and compare the result to the exponential density. Then write  $A_i = \sum_{k=1}^i X_k$ , and compute the moment-generating function for  $A_i$  and use that the inter-arrival times  $X_i$  are independent. Use the moment-generating function of  $X_i$ .

**h.1.3.12.** Use the standard integral  $\int_0^\infty x^n e^{-\alpha x} dx = \alpha^{-n-1} n!$ . Another way would be to use that, once you have the moment-generating function of some random variable  $X$ ,  $E[X] = \frac{d}{dt} M_X(t)|_{t=0}$ .

**h.1.3.13.** Realize that  $P(N(t) = k) = P(A_k \leq t) - P(A_{k+1} \leq t)$ .

**h.1.3.14.** Condition on the event  $X > t$ .

**h.1.3.15.** Use that if  $Z = \min\{X, S\} > x$ , it then must be that  $X > x$  and  $S > x$ . Then use independence of  $X$  and  $S$ .

**h.1.3.16.** Define the joint distribution of  $X$  and  $S$  and carry out the computations, or use conditioning, or use the result of the previous exercise.



*Solutions*

**s.1.3.1.** If  $\lambda < 0$ , then  $1 - e^{-\lambda t}$  grows to  $-\infty$  if  $t \rightarrow \infty$ . Just by itself this is not a problem. However, in our case  $1 - e^{-\lambda t}$  has the interpretation of a distribution function. Now, recall that a distribution function is bounded to values in the interval  $[0, 1]$ .

Suppose  $\lambda = 0$ . Then  $P(X \leq t) = 1 - e^0 = 0$ . In words, this would mean that the probability of a finite inter-arrival time between any two customers is zero. So, no customers can arrive in this case.

**s.1.3.2.**

$$\begin{aligned} E[X] &= \int_0^\infty t \lambda e^{-\lambda t} dt, \quad \text{density is } \lambda e^{-\lambda t} \\ &= \lambda^{-1} \int_0^\infty u e^{-u} du, \quad \text{by change of variable } u = \lambda t, \\ &= -\lambda^{-1} u e^{-u} \Big|_0^\infty + \lambda^{-1} \int_0^\infty e^{-u} du \\ &= -\lambda^{-1} e^{-u} \Big|_0^\infty = \frac{1}{\lambda}. \end{aligned}$$

**s.1.3.3.**

$$\begin{aligned} E[X^2] &= \int_0^\infty t^2 \lambda e^{-\lambda t} dt \\ &= \lambda^{-2} \int_0^\infty u^2 e^{-u} du, \quad \text{by change of variable } u = \lambda t, \\ &= -\lambda^{-2} t^2 e^{-t} \Big|_0^\infty + 2\lambda^{-2} \int_0^\infty t e^{-t} dt \\ &= -2\lambda^{-2} t e^{-t} \Big|_0^\infty + 2\lambda^{-2} \int_0^\infty e^{-t} dt \\ &= -2\lambda^{-2} e^{-t} \Big|_0^\infty \\ &= 2/\lambda^2. \end{aligned}$$

**s.1.3.4.** By the previous problems,  $E[X^2] = 2/\lambda^2$  and  $E[X] = 1/\lambda$ . Hence the variance  $V[X] = E[X^2] - (E[X])^2 = \frac{2}{\lambda^2} - (\frac{1}{\lambda})^2 = \frac{1}{\lambda^2}$ .

**s.1.3.5.**

$$\begin{aligned} M_X(t) &= E[e^{tX}] = \int_0^\infty e^{tx} f(x) dx \\ &= \lambda \int_0^\infty e^{tx} e^{-\lambda x} dx = \lambda \int_0^\infty e^{-(\lambda-t)x} dx. \end{aligned}$$

This last integral only converges when  $\lambda - t > 0$ .

**s.1.3.6.**

$$\begin{aligned} M_X(t) &= E[\exp(tX)] = \int_0^\infty e^{tx} f(x) dx = \int_0^\infty e^{tx} \lambda e^{-\lambda x} dx \\ &= \lambda \int_0^\infty e^{(t-\lambda)x} dx = \frac{\lambda}{\lambda - t}. \end{aligned}$$

**s.1.3.7.** By [1.3.6],  $M'_X(t) = \lambda/(\lambda - t)^2$ . Hence,  $M'_X(0) = 1/\lambda$ . Next,  $M''_X(t) = 2\lambda/(\lambda - t)^3$ , hence  $E[X^2] = M''_X(0) = 2\lambda/\lambda^3 = 2\lambda^{-2}$ .

**s.1.3.8.** By the previous problems,  $V[X] = 1/\lambda^2$  and  $E[X] = 1/\lambda$ , hence

$$C^2 = \frac{V[X]}{(E[X])^2} = \frac{1/\lambda^2}{1/\lambda^2} = 1.$$

**s.1.3.9.** If there are no arrivals in some interval  $[0, t]$ , then it must be that  $N(t) = 0$ . Hence, for the first inter-arrival time  $X_1$ :

$$P(X_1 > t) = P(N(t) = 0) = e^{-\lambda t} \frac{(\lambda t)^0}{0!} = e^{-\lambda t}.$$

**s.1.3.10.** The simplest way to obtain the answer is to use that the expectation is a linear operator, i.e.,  $E[X + Y] = E[X] + E[Y]$  for any r.v.  $X$  and  $Y$ . Then,

$$E[A_i] = E\left[\sum_{k=1}^i X_k\right] = i E[X] = \frac{i}{\lambda}.$$

(Just as a reminder,  $E[XY] \neq E[X]E[Y]$  in general. Only when  $X$  and  $Y$  are uncorrelated (which is implied by independence), the product of the expectations is the expectation of the products.)

**s.1.3.11.** One way to find the distribution of  $A_i$  is by using the moment-generating function  $M_{A_i}(t) = E[e^{tA_i}]$  of  $A_i$ . Let  $X_i$  be the inter-arrival time between customers  $i$  and  $i - 1$ , and  $M_X(t)$  the associated moment-generating function. Using the i.i.d. property of the  $\{X_i\}$ ,

$$\begin{aligned} M_{A_i}(t) &= E[e^{tA_i}] = E\left[\exp\left(t \sum_{k=1}^i X_k\right)\right] \\ &= \prod_{k=1}^i E[e^{tX_k}] = \prod_{k=1}^i M_{X_k}(t) = \prod_{k=1}^i \frac{\lambda}{\lambda - t} = \left(\frac{\lambda}{\lambda - t}\right)^i. \end{aligned}$$

From a table of moment-generating functions it follows immediately that  $A_i \sim \Gamma(i, \lambda)$ , i.e.,  $A_i$  is Gamma distributed.

**s.1.3.12.**

$$E[A_i] = \int_0^\infty t f_{A_i}(t) dt = \int_0^\infty t \lambda e^{-\lambda t} \frac{(\lambda t)^{i-1}}{(i-1)!} dt.$$

Thus,

$$E[A_i] = \frac{1}{(i-1)!} \int_0^\infty e^{-\lambda t} (\lambda t)^i dt = \frac{i!}{(i-1)! \lambda} = \frac{i}{\lambda},$$

where we used the hint.

What if we would use the moment-generating function, as derived by the previous exercise?

$$\begin{aligned} E[A_i] &= \left. \frac{d}{dt} M_{A_i}(t) \right|_{t=0} \\ &= \left. \frac{d}{dt} \left( \frac{\lambda}{\lambda - t} \right)^i \right|_{t=0} \\ &= i \left( \frac{\lambda}{\lambda - t} \right)^{i-1} \frac{\lambda}{(\lambda - t)^2} \Big|_{t=0} = \frac{i}{\lambda}. \end{aligned}$$

**s.1.3.13.** We want to show that

$$P(N(t) = k) = e^{-\lambda t} \frac{(\lambda t)^k}{k!}.$$

Now observe that  $P(N(t) = k) = P(A_k \leq t) - P(A_{k+1} \leq t)$ . Using the density of  $A_{k+1}$  and applying partial integration leads to

$$\begin{aligned} P(A_{k+1} \leq t) &= \lambda \int_0^t \frac{(\lambda s)^k}{k!} e^{-\lambda s} ds \\ &= \lambda \frac{(\lambda s)^k}{k!} \frac{e^{-\lambda s}}{-\lambda} \Big|_0^t + \lambda \int_0^t \frac{(\lambda s)^{k-1}}{(k-1)!} e^{-\lambda s} ds \\ &= -\frac{(\lambda t)^k}{k!} e^{-\lambda t} + P(A_k \leq t). \end{aligned}$$

We are done.

**s.1.3.14.** By the definition of conditional probability

$$P(X > t+h | X > t) = \frac{P(X > t+h, X > t)}{P(X > t)} = \frac{P(X > t+h)}{P(X > t)} = \frac{e^{-\lambda(t+h)}}{e^{-\lambda t}} = e^{-\lambda h} = P(X > h).$$

Thus, no matter how long we have been waiting for the next arrival to occur, the probability that it will occur in the next  $h$  seconds remains the same.

This property seems to be also vindicated in practice: suppose that a patient with a broken arm just arrived at the emergency room of a hospital, what does that tell us about the time the next patient will be brought in? Not much, as most of us will agree.

**s.1.3.15.** Use that  $X$  and  $S$  are independent to get

$$\begin{aligned} P(Z > x) &= P(\min\{X, S\} > x) = P(X > x \text{ and } S > x) = P(X > x)P(S > x) \\ &= e^{-\lambda x} e^{-\mu x} = e^{-(\lambda+\mu)x}. \end{aligned}$$

**s.1.3.16.** There is more than one way to show that  $P(X \leq S) = \lambda/(\lambda + \mu)$ .

Observe first that  $X$  and  $S$ , being exponentially distributed, have a density. Moreover, as they are independent, the joint density takes the form

$$f_{X,S}(x, y) = f_X(x)f_S(y) = \lambda\mu e^{-\lambda x} e^{-\mu y}.$$

With this,

$$\begin{aligned} P(X \leq S) &= E[\mathbb{1}_{X \leq S}] \\ &= \int_0^\infty \int_0^\infty \mathbb{1}_{x \leq y} f_{X,S}(x, y) dy dx \\ &= \lambda\mu \int_0^\infty \int_0^\infty \mathbb{1}_{x \leq y} e^{-\lambda x} e^{-\mu y} dy dx \\ &= \lambda\mu \int_0^\infty e^{-\mu y} \int_0^y e^{-\lambda x} dx dy \\ &= \mu \int_0^\infty e^{-\mu y} (1 - e^{-\lambda y}) dy \\ &= \mu \int_0^\infty (e^{-\mu y} - e^{-(\lambda+\mu)y}) dy \\ &= 1 - \frac{\mu}{\lambda + \mu} \\ &= \frac{\lambda}{\lambda + \mu}. \end{aligned}$$

#### 1.4 CONSTRUCTION OF THE SINGLE-SERVER QUEUEING PROCESS IN CONTINUOUS TIME

In Section 1.2 we modeled time as progressing in discrete ‘chunks’: minutes, hours, days, and so on. For given numbers of arrivals and capacity per period, we use the recursions (1.2.1) to compute the departures and queue length per period. However, we can also model time in a continuous way, so that jobs can arrive at any moment and have arbitrary service times. In this section, we consider a single-server FIFO queueing process in continuous time.

##### *Theory and Exercises*

Assume we are given the *arrival process*  $\{A(t); t \geq 0\}$ , i.e., the number of jobs that arrived during  $[0, t]$ . Thus,  $\{A(t); t \geq 0\}$  is a *counting process*.

From this arrival process, we can derive various other interesting concepts, such as the *arrival times* of individual jobs. Especially, if we know that  $A(s) = k - 1$  and  $A(t) = k$ , then the arrival time  $A_k$  of the  $k$ th job must lie somewhere in  $(s, t]$ . Thus, from  $\{A(t)\}$ , we can define<sup>2</sup>

$$A_k = \min\{t : A(t) \geq k\}, \quad (1.4.1)$$

and set  $A_0 = 0$ . Once we have the set of arrival times  $\{A_k\}$ , the set of *inter-arrival times*  $\{X_k, k = 1, 2, \dots\}$  between consecutive customers can be constructed as

$$X_k = A_k - A_{k-1}. \quad (1.4.2)$$

However, often the basic data consists of the inter-arrival times  $\{X_k; k = 1, 2, \dots\}$  rather than the arrival times  $\{A_k\}$  or the arrival process  $\{A(t)\}$ . Then we construct the arrival times as

$$A_k = A_{k-1} + X_k,$$

with  $A_0 = 0$ . From the arrival times  $\{A_k\}$  we can, in turn, construct the arrival process  $\{A(t)\}$  as

$$A(t) = \max\{k : A_k \leq t\}. \quad (1.4.3)$$

Thus, from the inter-arrival times  $\{X_k\}$  it is possible to construct  $\{A_k\}$  and  $\{A(t)\}$ , and from  $\{A(t)\}$  we can obtain  $\{A_k\}$  and  $\{X_k\}$ .

**1.4.1.** Show that we can also define  $A(t)$  as

$$A(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t}. \quad (1.4.4)$$

Thus, we just count all arrivals that occur up to and including time  $t$ .

**1.4.2.** Is it true that the following mappings are correct:

$$\begin{aligned} A_k : \mathbb{N} &\rightarrow \mathbb{R}, & \text{job id (integer) to arrival time (real number),} \\ A(t) : \mathbb{R} &\rightarrow \mathbb{N}, & \text{time (real number) to number of jobs (integer)?} \end{aligned}$$

**1.4.3.** What are the meanings of  $A_{A(t)}$  and  $A(A_n)$ ?

<sup>2</sup> If we want to be mathematically precise, we must take  $\inf$  rather than  $\min$ . However, in this book we do not want to distinguish between subtleties.



Figure 3: Construction of the single-server queue in continuous time. The sojourn time  $W_k$  of the  $k$ th job is the sum of the work in queue  $W_{Q,k}$  at its arrival epoch  $A_k$  and its service time  $S_k$ ; its departure time is then  $D_k = A_k + W_k$ . The waiting time of job  $k$  is clearly equal to  $W_{k-1} - X_k$ . We also see that job  $k+1$  arrives at an empty system, hence its sojourn time  $W_{k+1} = S_{k+1}$ . Finally, the virtual waiting time process is shown by the lines with slope  $-1$ .

**1.4.4.** In view of the above, can  $A(t)$  be defined as  $\min\{k : A_k \geq t\}$  or as  $\min\{k : A_k > t\}$ ?

The *service times* of the jobs are given by the sequence  $\{S_k\}$ . Typically we assume that the service times are i.i.d., and also independent of the arrival process  $\{A(t)\}$ . With the arrival times and service times we can construct the *waiting time in queue*  $\{W_{Q,k}\}$  as seen by the jobs at the moment they arrive. From Fig. 3 it is evident that

$$W_{Q,k} = [W_{Q,k-1} + S_{k-1} - X_k]^+. \quad (1.4.5)$$

With this it is easy to compute the waiting times: from the initial condition  $W_{Q,0} = 0$  we can obtain  $W_{Q,1}$ , and then  $W_{Q,2}$ , and so on.

**1.4.5.** Explain that  $W_{Q,k} = [W_{Q,k-1} + S_{k-1} - X_k]^+$ .

**1.4.6.** If  $S \sim U[0, 7]$  and  $X \sim U[0, 10]$ , where  $U[I]$  stands for the uniform distribution concentrated on the interval  $I$ , compute  $P(S - X \leq u)$ , for  $S$  and  $X$  independent.

The time job  $k$  leaves the queue and moves to the server is given by

$$\tilde{A}_k = A_k + W_{Q,k},$$

because a job can only move to the server after its arrival plus the time it needs to wait in queue. Note that we here explicitly use the FIFO assumption. Right after the job moves from the queue to the server, its service starts. Thus,  $\tilde{A}_k$  is also the epoch at which the service of job  $k$  starts.

After completing its service, the job leaves the system. Hence, the *departure time of the system* is

$$D_k = \tilde{A}_k + S_k.$$

This in turn specifies the departure process  $\{D(t)\}$  as

$$D(t) = \max\{k : D_k \leq t\} = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \leq t}.$$

The *sojourn time*, or *waiting time in the system*,  $W_k$ , is the time a job spends in the entire system. With the above relations we see that

$$W_k = D_k - A_k = \tilde{A}_k + S_k - A_k = W_{Q,k} + S_k, \quad (1.4.6)$$

where each of these equations has its own interpretation.

**1.4.7.** Explain that the following recursion for  $W_k$  is valid:

$$W_{Q,k} = [W_{k-1} - X_k]^+, \quad W_k = W_{Q,k} + S_k = [W_{k-1} - X_k]^+ + S_k, \quad (1.4.7)$$

from which follows a recursion for  $D_k$ :

$$D_k = A_k + W_k. \quad (1.4.8)$$

**1.4.8.** Assume that  $X_1 = 10$ ,  $X_2 = 5$ ,  $X_3 = 6$  and  $S_1 = 17$ ,  $S_2 = 20$  and  $S_3 = 5$ , compute the arrival times, waiting times in queue, the sojourn times and the departure times for these three customers.

**1.4.9.** Assume that  $X_k = 10$  minutes and  $S_k = 11$  minutes for all  $k$ , i.e.,  $X_k$  and  $S_k$  are deterministic and constant. What are  $\lambda$  and  $\mu$ ? Compute  $A_k$ ,  $W_k$ ,  $D_k$  as functions of  $k$ . Then find expressions for  $A(t)$  and  $D(t)$ .

**1.4.10.** Suppose that  $X_k \in \{1, 3\}$  such that  $P(X_k = 1) = P(X_k = 3)$  and  $S_k \in \{1, 2\}$  with  $P(S_k = 1) = P(S_k = 2)$ . If  $W_{Q,0} = 3$ , what are the distributions of  $W_{Q,1}$  and  $W_{Q,2}$ ?

**1.4.11.** Explain the following recursions for a single server queue:

$$\begin{aligned} A_k &= A_{k-1} + X_k, \\ D_k &= \max\{A_k, D_{k-1}\} + S_k, \\ W_k &= D_k - A_k. \end{aligned} \quad (1.4.9)$$

The *virtual waiting time process*  $\{V(t)\}$  is the amount of waiting that an arrival would see if it would arrive at time  $t$ . To construct  $\{V(t)\}$ , we simply draw lines that start at points  $(A_k, W_k)$  and have slope  $-1$ , unless the line hits the  $x$ -axis, in which case the virtual waiting time remains zero until the next arrival occurs.

**1.4.12.** Provide a specification of the virtual waiting time process  $\{V(t)\}$  for all  $t$ .

Once we have the arrival and departure processes it is easy to compute the *number of jobs in the system* at time  $t$  as, cf. Fig. 4,

$$L(t) = A(t) - D(t) + L(0), \quad (1.4.10)$$

where  $L(0)$  is the number of jobs in the system at time  $t = 0$ ; typically we assume that  $L(0) = 0$ . As in a queueing system, jobs can be in queue or in service, we distinguish between the number in the system  $L(t)$ , the number in queue  $L_Q(t)$ , and the number of jobs in service  $L_s(t)$ .

**1.4.13.** In [1.4.9], find an expression for  $L(A_k-)$ .

In summary, starting from a sequence of inter-arrival times  $\{X_k\}$  and service times  $\{S_k\}$  we can obtain a set of recursions by which we can simulate a queueing process in continuous time. A bit of experimentation with Python (or R perhaps) will reveal that this is easy.

**1.4.14.** Explain that if we know  $\tilde{A}(t)$ , i.e. the number of jobs that departed from the queue up to time  $t$ , then

$$\begin{aligned} L_Q(t) &= A(t) - \tilde{A}(t), \\ L_s(t) &= \tilde{A}(t) - D(t) = L(t) - L_Q(t). \end{aligned}$$

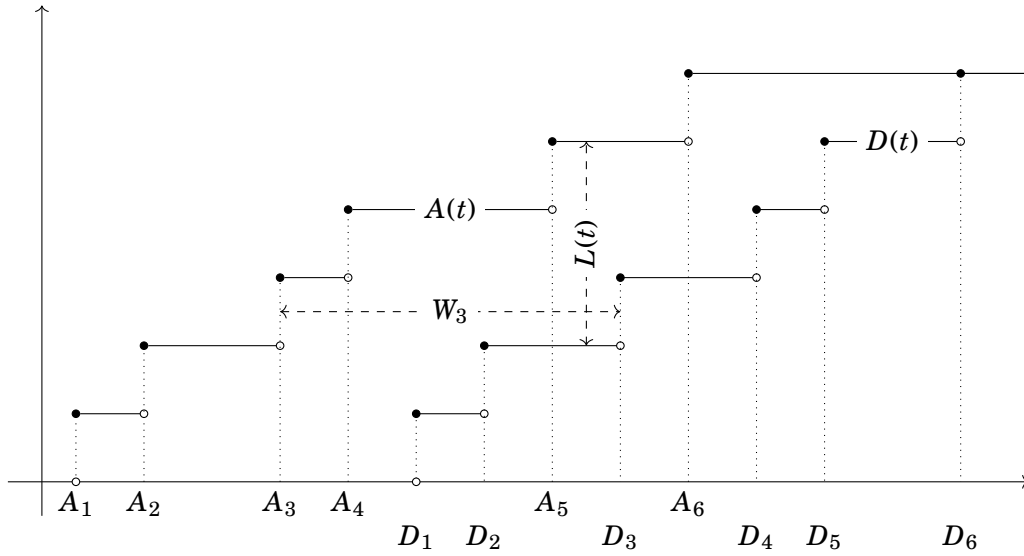


Figure 4: Relation between the arrival process  $\{A(t)\}$ , the departure process  $\{D(t)\}$ , the number in the system  $\{L(t)\}$  and the waiting times  $\{W_k\}$ . In particular,  $L(t)$  is the difference between the graphs of  $A(t)$  and  $D(t)$ .

**1.4.15.** Why don't we need a separate notation for  $D_s(t)$ , the number of jobs that departed from the server?

**1.4.16.** Is  $\tilde{A}(t) \leq D(t)$  or  $\tilde{A}(t) \geq D(t)$ ?

**1.4.17.** Consider a multi-server queue with  $m$  servers. Suppose that at some time  $t$  it happens that  $\tilde{A}(t) - D(t) < m$  even though  $A(t) - D(t) > m$ . How can this occur?

**1.4.18.** Show that  $L(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t < D_k}$  when the system starts empty.

**1.4.19.** Show that  $L(A_k) > 0 \implies A_k \leq D_{k-1}$ .

Define the number of jobs in the system as seen by the  $k$ th arrival as

$$L(A_k-). \tag{1.4.11}$$

Observe that we write  $A_k-$ , and not  $A_k$ . As we are concerned with a process with jumps, we need to be quite particular about left and right limits at jump epochs.

**1.4.20.** With the recursions (1.4.7) it is apparently easy to compute the waiting time (in queue), but it is less simple to compute the number of jobs in queue or in the system. In this exercise we develop an algorithm to compute the number of jobs in the system as seen by arrivals. Explain why the following (algorithmic efficient) procedure works:

$$L(A_k-) = L(A_{k-1}-) + 1 - \sum_{i=k-1-L(A_{k-1}-)}^{k-1} \mathbb{1}_{D_i < A_k}.$$

**1.4.21.** In [1.4.20], why do we take  $i = k - 1 - L(A_{k-1}-)$  in the sum, and not  $i = k - 2 - L(A_{k-1}-)$ ?

**1.4.22** (Not obligatory). Try to extend the recursions (1.4.9) to a situation in which one queue is served by two servers.

*Hints*

**h.1.4.1.** What is  $\mathbb{1}_{A_k \leq t}$  if  $A_k \leq t$ ?

**h.1.4.4.** Compare this to the definition in (1.4.3).

**h.1.4.6.** This is elementary, hence it might appear trivial, but it's not. . . In fact, I had a hard time finding a simple way to get the answer. It is good practice to try yourself before looking at the answer. Check also the previous problem, and make a drawing of the region over which you have to integrate.

**h.1.4.8.** The intent of this exercise is to make you familiar with the notation.

BTW, such simple test cases are also very useful to test computer code. The numbers in the exercise are one such simple case. You can check the results by hand; if the results of the simulator are different, there is a problem.

**h.1.4.10.** Use (1.4.7).

**h.1.4.12.** Make a plot of the function  $A_{A(t)} - t$ . What is the meaning of  $V(A_{A(t)})$ ? What is  $V(A_{A(t)}) + A_{A(t)} - t$ ?

**h.1.4.18.** Use Boolean algebra. Write, for notational ease,  $A = \mathbb{1}_{A_k \leq t}$  and  $\bar{A} = 1 - A = \mathbb{1}_{A_k > t}$ , and define something similar for  $D$ . Then show that  $A - D = A\bar{D} - \bar{A}D$ , and show that  $\bar{A}D = 0$ . Finally sum over  $k$ .

**h.1.4.19.** Use that  $L(A_k) > 0$  means that the system contains at least one job at the time of the  $k$ th arrival, and that  $A_k \leq D_{k-1}$  means that job  $k$  arrives before job  $k - 1$  departs.

**h.1.4.22.** The problem is that in a multi-server queueing systems, unlike for single-server queues, jobs can overtake each other: a small job that arrives after a very large job can still leave the system earlier. After trying for several hours, I obtained an inelegant method. A subsequent search on the web helped a lot. The solution below is a modification of N. Krivulin, 'Recursive equations based models of queueing systems'.



### Solutions

**s.1.4.1.** For every  $A_k \leq t$ , we have that  $\mathbb{1}_{A_k \leq t} = 1$ , and else the indicator is 0. Hence, in the summation we count the number of times  $A_k \leq t$ .

**s.1.4.2.** Yes, it is true.

**s.1.4.3.**  $A(t)$  is the number of arrivals during  $[0, t]$ . Suppose that  $A(t) = n$ . This  $n$ th job arrived at time  $A_n$ . Thus,  $A_{A(t)}$  is the arrival time of the last job that arrived before or at time  $t$ . In a similar vein,  $A_n$  is the arrival time of the  $n$ th job. Thus, the number of arrivals up to time  $A_n$ , i.e.,  $A(A_n)$ , must be  $n$ .

**s.1.4.4.** Suppose  $A_3 = 10$  and  $A_4 = 20$ . Take  $t = 15$ . Then  $\min\{k : A_k \geq 15\} = 4$  since  $A_3 < t = 15 < A_4$ . On the other hand  $\max\{k : A_k \leq t\} = 3$ . And, indeed, at time  $t = 15$ , 3 jobs arrived, not 4. So defining  $A(t)$  as  $\min\{k : A_k \geq t\}$  is not OK. This example also shows that in general  $A(t) \neq \min\{k : A_k > t\}$ . So, neither definition is correct.

**s.1.4.5.** The waiting time of the  $k$ th arrival must be equal to the waiting time of the  $k - 1$ th customer plus the amount of *service time* required by job  $k - 1$  minus the time that elapses between the arrival of job  $k - 1$  and job  $k$ , unless the server becomes idle between jobs  $k - 1$  and  $k$ .

**s.1.4.6.** The joint density of  $S$  and  $X$  is given by

$$f_{XS}(x, s) = f_X(x) \cdot f_S(s) = \frac{1}{10} \mathbb{1}_{0 \leq x \leq 10} \cdot \frac{1}{7} \mathbb{1}_{0 \leq s \leq 7},$$

since  $X$  and  $S$  are independent. Thus,

$$\begin{aligned} P(S - X \leq u) &= E[\mathbb{1}_{S - X \leq u}] = \frac{1}{70} \int_0^{10} \int_0^7 \mathbb{1}_{s - x \leq u} \, ds \, dx \\ &= \frac{1}{70} \int_0^{10} \int_0^7 \mathbb{1}_{s \leq x + u} \, ds \, dx. \end{aligned}$$

Now we need to chop up the domain of  $P(S - X \leq u)$ , for which we use the figure below.



It is clear that the indicated rectangle has no overlap with the set of points  $(x, s)$  such that  $s \leq u + x$  for  $u < -10$ . (To see this, draw the line  $s = x - 10$  in the figure.) At  $u = -10$ , the overlap is a single point, at  $(10, 0)$ . Thus,

$$P(S - X \leq u) = 0, \quad \text{for } u \leq -10.$$

When  $u \in [-10, -3]$  we need to integrate over the triangle that results from cutting the line  $s = x + u$  with the rectangle. The area is

$$70P(S - X \leq u) = \frac{(10 + u)^2}{2}, \quad \text{for } -10 \leq u \leq -3,$$

where we multiply with 70 to get the normalization right.

When  $u \in [-3, 0]$ , we integrate over a parallelogram with base  $3 + u$  and height 7 plus the triangle below the line  $s = x - 3$ . The area is

$$70P(S - X \leq u) = (3 + u)7 + \frac{(10 - 3)^2}{2} = 7u + \frac{91}{2}, \quad \text{for } -3 \leq u \leq 0.$$

For  $u \in [0, 7]$ , we integrate over the trapezoid that results from intersecting the set  $\{(x, s) : x \leq s \leq s + u\}$  and the rectangle plus the parallelogram plus the triangle below the line  $s = x - 3$ . The area is

$$70P(S - X \leq u) = \frac{7^2}{2} - \frac{(7 - u)^2}{2} + 3 \cdot 7 + \frac{49}{2} = 7u - \frac{u^2}{2} + \frac{91}{2}, \quad \text{for } 0 \leq u \leq 7.$$

Finally, for  $u \geq 7$ , the set  $s \leq x + u$  covers the entire rectangle. Hence,

$$70P(S - X \leq u) = 70, \quad \text{for } 7 \leq u.$$

Given the amount of effort I had to put into getting this answer, I wanted to check it. So I went to Wolfram Alpha (which is a great site for symbolic computations), and typed this:

```
\int_{0}^{10} \int_{0}^7 Boole[s <= x + u] ds dx,
```

so, once you know ~~LaTeX~~ you can use Wolfram Alpha. Wolfram Alpha turned it to

```
Integrate[Boole[s <= u + x], {x, 0, 10}, {s, 0, 7}]
```

If you fill this in at Wolfram, you'll get the results that we obtained above in seconds, rather than in one hour or so (depending on your proficiency with carrying out integrals).

**s.1.4.7.** Suppose  $k$  arrives right after job  $k - 1$ . Then job  $k$  has to wait until job  $k - 1$  leaves, in other words, the service of job  $k$  can start only after job  $k - 1$  leaves. Consequently, in this case,  $W_{Q,k} = W_{k-1}$ . If there is time  $X_k$  between jobs  $k$  and  $k - 1$ , then job  $k$  has to spend  $X_k$  less in queue. This explains the first formula. The rest follows right away.

**s.1.4.8.** Let's feed it to the computer. Mind that in Python (just like in C, and so on), arrays start at index 0, not at index 1.

```
>>> X = [0, 10, 5, 6]
>>> S = [0, 17, 20, 5]
>>> A = [0, 0, 0, 0]
>>> for i in range(1, len(X)):
...     A[i] = A[i-1] + X[i]
...
>>> A
[0, 10, 15, 21]

>>> WQ = [0, 0, 0, 0]
>>> for i in range(1, len(X)):
...     WQ[i] = max(WQ[i-1] + S[i-1] - X[i], 0)
...
>>> WQ
[0, 0, 12, 26]

>>> ST = [0,0,0,0]
>>> for i in range(1, len(X)):
...     ST[i] = WQ[i] + S[i]
...
>>> ST
[0, 17, 32, 31]

>>> D = [0, 0, 0, 0]
>>> for i in range(1, len(X)):
...     D[i] = A[i] + WQ[i] + S[i]
```

...  
 >>> D  
 [0, 27, 47, 52]

**s.1.4.9.**  $\lambda = 6$  per hour, and  $\mu = 60/11$  per hour. Note that  $\mu < \lambda$ .  $A_0 = 0$ ,  $A_1 = 10$ ,  $A_2 = 20$ , and so on. Hence,  $A_k = 10k$ .  $W_{Q,0} = 0$ ,  $W_{Q,1} = \max\{0 + 0 - 10, 0\} = 0$ .  $W_{Q,2} = \max\{0 + 11 - 10, 0\} = 1$ .  $W_{Q,3} = \max\{1 + 11 - 10, 0\} = 2$ . Hence,  $W_{Q,k} = k - 1$  for  $k \geq 1$ . Thus,  $W_k = k - 1 + 11 = k + 10$  for  $k \geq 1$ , and  $D_k = 10k + k + 10 = 11k + 10$ . Note that  $W_k$  increases linearly as a function of  $k$ . All in all,  $A(t) = \lfloor t/10 \rfloor$ , and  $D(t) = \lfloor (t - 10)/11 \rfloor$ .

**s.1.4.10.** First find the distribution of  $Y_k := S_{k-1} - X_k$  so that we can write  $W_{Q,k} = [W_{Q,k-1} + Y_k]^+$ . Use independence of  $\{S_k\}$  and  $\{X_k\}$ :

$$P(Y_k = -2) = P(S_{k-1} - X_k = -2) = P(S_{k-1} = 1, X_k = 3) = P(S_{k-1} = 1)P(X_k = 3) = \frac{1}{4}.$$

Dropping the dependence on  $k$  for ease, we get

$$P(Y = -2) = P(S - X = -2) = P(S = 1, X = 3) = P(S = 1)P(X = 3) = \frac{1}{4},$$

$$P(Y = -1) = P(S = 2)P(X = 3) = \frac{1}{4},$$

$$P(Y = 0) = P(S = 1)P(X = 1) = \frac{1}{4},$$

$$P(Y = 1) = P(S = 2)P(X = 1) = \frac{1}{4}.$$

With this

$$P(W_{Q,1} = 1) = P(W_{Q,0} + Y = 1) = P(0 + Y = 1) = P(Y = -2) = \frac{1}{4},$$

$$P(W_{Q,1} = 2) = P(0 + Y = 2) = P(Y = -1) = \frac{1}{4},$$

$$P(W_{Q,1} = 3) = P(0 + Y = 3) = P(Y = 0) = \frac{1}{4},$$

$$P(W_{Q,1} = 4) = P(0 + Y = 4) = P(Y = 1) = \frac{1}{4}.$$

And, then

$$\begin{aligned} P(W_{Q,2} = 1) &= P(W_{Q,1} + Y = 1) = \sum_{i=1}^4 P(W_{Q,1} + Y = 1 | W_{Q,1} = i) P(W_{Q,1} = i) \\ &= \sum_{i=1}^4 P(i + Y = 1 | W_{Q,1} = i) \frac{1}{4} = \sum_{i=1}^4 P(Y = 1 - i | W_{Q,1} = i) \frac{1}{4} \\ &= \frac{1}{4} \sum_{i=1}^4 P(Y = 1 - i) = \frac{1}{4} (P(Y = 0) + P(Y = -1) + P(Y = -2)) = \frac{3}{16}. \end{aligned}$$

**s.1.4.11.** Of course, the service of job  $k$  cannot start before it arrives. Hence, it cannot leave before  $A_k + S_k$ . Therefore it must be that  $D_k \geq A_k + S_k$ . But the service of job  $k$  can also not start before the previous job, i.e. job  $k - 1$ , left the server. Thus job  $k$  cannot start before  $D_{k-1}$ . To clarify it somewhat further, define  $S'_k$  as the earliest start of job  $k$ . Then it must be that  $S'_k = \max\{A_k, D_{k-1}\}$ —don't confuse the earliest start  $S'_k$  and the service time  $S_k$ —and  $D_k = S'_k + S_k$ .

**s.1.4.12.** There is a funny way to do this. Recall from a previous exercise that if  $A(t) = n$ , then  $A_n$  is the arrival time of the  $n$ th job. Thus, the function  $A_{A(t)}$  provides us with arrival times as a function of  $t$ . When  $t = A_{A(t)}$ , i.e., when  $t$  is the arrival time of the  $A(t)$ th job, we set  $V(t) = V(A_{A(t)}) = W_{A(t)}$ , i.e., the virtual waiting time at the arrival time  $t = A_{A(t)}$  is equal to the waiting time of the  $A(t)$ th job. Between arrival moments, the virtual waiting time decreases with slope 1, until it hits 0. Thus,

$$V(t) = [V(A_{A(t)}) - (t - A_{A(t)})]^+ = [W_{A(t)} + (A_{A(t)} - t)]^+.$$

The notation may be a bit confusing, but it is in fact very simple. Take some  $t$ , look back at the last arrival time before time  $t$ , which is written as  $A_{A(t)}$ . (In computer code these times are easy to find.) Then draw a line with slope  $-1$  from the waiting time that the last arrival saw.

**s.1.4.13.** Recall that  $A(t) = \lfloor t/10 \rfloor$ , and  $D(t) = \lfloor (t - 10)/11 \rfloor$ . Hence,

$$L(A_k -) = k - 1 - D(A_k -) = k - 1 - D(10k -) = k - 1 - \left\lfloor \frac{(10k -) - 10}{11} \right\rfloor.$$

The computation is a bit tricky since sometimes arrivals and departures coincide. (Consider for instance  $t = 120$ .)

From this example you can infer that it is necessary that the service rate is greater than the arrival rate, i.e.,  $\mu > \lambda$ , for otherwise, the queue length keeps on increasing (in the long run).

**s.1.4.14.** We defined  $\tilde{A}(t)$  as the amount of people up to time  $t$  that left the queue and moved to the server. It is then clear to see that the number of jobs in queue  $L_Q(t)$  is equal to the number amount of jobs that have arrived, i.e.,  $A(t)$ , minus the number of jobs that left the queue, i.e.,  $\tilde{A}(t)$ . Using the same reasoning for  $L_s(t)$ , the second line also follows.

**s.1.4.15.** Because  $D_s(t) = D(t)$ . Once customers leave the server, their service is completed, and they leave the queueing system.

**s.1.4.16.** All customers that left the system must have left the queue. Thus,  $\tilde{A}(t) \geq D(t)$ .

**s.1.4.17.** In this case, there are servers idling while there are still customers in queue. If such events occur, we say that the server is not work-conservative.

**s.1.4.18.**

$$\begin{aligned} L(t) &= A(t) - D(t) \\ &= \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t} - \sum_{k=1}^{\infty} \mathbb{1}_{D_k \leq t} \\ &= \sum_{k=1}^{\infty} [\mathbb{1}_{A_k \leq t} - \mathbb{1}_{D_k \leq t}]. \end{aligned}$$

Write for the moment  $A = \mathbb{1}_{A_k \leq t}$  and  $\bar{A} = 1 - A = \mathbb{1}_{A_k > t}$ , and likewise for  $D$ . Now we can use Boolean algebra to see that  $\mathbb{1}_{A_k \leq t} - \mathbb{1}_{D_k \leq t} = A - D = A(D + \bar{D}) - D = AD + A\bar{D} - D = A\bar{D} - D(1 - A) = A\bar{D} - D\bar{A}$ . But  $D\bar{A} = 0$  since  $D\bar{A} = \mathbb{1}_{D_k \leq t} \mathbb{1}_{A_k > t} = \mathbb{1}_{D_k \leq t < A_k}$  which would mean that the arrival time  $A_k$  of the  $k$ th job would be larger than its departure time  $D_k$ . As  $A\bar{D} = \mathbb{1}_{A_k \leq t < D_k}$

$$\begin{aligned} L(t) &= \sum_{k=1}^{\infty} [\mathbb{1}_{A_k \leq t} - \mathbb{1}_{D_k \leq t}] \\ &= \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t < D_k}. \end{aligned}$$

Boolean algebra is actually a really nice way to solve logical puzzles. If you are interested, you can find some examples on my homepage.

**s.1.4.19.** In a sense, the claim is evident, for, if the system contains a job when job  $k$  arrives, it cannot be empty. But if it is not empty, then at least the last job that arrived before job  $k$ , i.e., job  $k-1$ , must still be in the system. That is,  $D_{k-1} \geq A_k$ . A more formal proof proceeds along the following lines. Using that  $A(A_k) = k$  and  $D(D_{k-1}) = k-1$ ,

$$\begin{aligned} L(A_k) > 0 &\Leftrightarrow A(A_k) - D(A_k) > 0 \Leftrightarrow k - D(A_k) > 0 \Leftrightarrow k > D(A_k) \\ &\Leftrightarrow k-1 \geq D(A_k) \Leftrightarrow D(D_{k-1}) \geq D(A_k) \Leftrightarrow D_{k-1} \geq A_k, \end{aligned}$$

where the last relation follows from the fact that  $D(t)$  is a counting process, hence monotone non-decreasing.

**s.1.4.20.** Let  $L(A_k-)$ , i.e., the number of jobs in the system as ‘seen by’ job  $k$ . It must be that  $L(A_k-) = k-1 - D(A_k)$ . To see this, assume first that no job has departed when job  $k$  arrives. Then job  $k$  must see  $k-1$  jobs in the system. In general, if at time  $A_k$  the number of departures is  $D(A_k)$ , then the above relation for  $L(A_k-)$  must hold. Applying this to job  $k-1$  we get that  $L(A_{k-1}-) = k-2 - D(A_{k-1})$ .

For the computation of  $L(A_k-)$  we do not have to take the departures before  $A_{k-1}$  into account as these have already been ‘incorporated in’  $L(A_{k-1}-)$ . Therefore,

$$L(A_k-) = L(A_{k-1}-) + 1 - \sum_{i=k-1-L(A_{k-1}-)}^{k-1} \mathbb{I}_{D_i < A_k}.$$

Suppose  $L(A_{k-1}) = 0$ , i.e., job  $k-1$  finds an empty system at its arrival and  $D_{k-1} > A_k$ , i.e., job  $k-1$  is still in the system when job  $k$  arrives. In this case,  $L(A_k-) = 1$ , which checks with the formula. Also, if  $L(A_{k-1}-) = 0$  and  $D_{k-1} < A_k$  then  $L(A_k-) = 0$ . This also checks with the formula.

**s.1.4.21.** The reason to start at  $k-1-L(A_{k-1}-)$  is that the number in the system as seen by job  $k$  is  $k-1-D(A_k)$  (not  $k-2-D(A_k)$ ). Hence, the jobs with index from  $k-1-L(A_{k-1}-)$ ,  $k-L(A_{k-1}-)$ ,  $\dots$ ,  $k-1$ , could have left the system between the arrival of job  $k-1$  and job  $k$ .

**s.1.4.22.** The recursions for the two-server system are this:

$$\begin{aligned} A_k &= A_{k-1} + X_k, \\ C_k &= \max\{A_k, D_{k-2}\} + S_k, \\ M_k &= \max\{M_{k-1}, C_k\}, \\ D_{k-1} &= \min\{M_{k-1}, C_k\}. \end{aligned}$$

Here,  $C_k$  is the completion time of job  $k$ , and  $\{D_k\}$  is a sorted list of departure times. Thus,  $D_k$  is the  $k$ th departure time; recall this is not necessarily equal to the completion time  $C_k$  of the  $k$ th job (as jobs may overtake each other). To understand the other equations, we reason like this. By construction,  $C_k > D_{k-m}$  (as  $S_k > 0$ ). Therefore, when we arrived at time  $C_k$ ,  $(k-m)$  jobs must have departed. Moreover, by construction,  $M_k$  tracks the latest completion time of all  $k$  jobs, hence,  $M_{k-m+1}$  is the latest completion time of the first  $k-m+1$  jobs. Therefore, if  $C_k > M_{k-1}$ , job  $k$  must leave later than the latest of the jobs in  $\{1, 2, \dots, k-1\}$ . Hence, the latest departure time of the jobs in  $\{1, 2, \dots, k-1\}$  jobs must be  $M_{k-1}$ . If however,  $C_k < M_{k-1}$ , then

job  $k$  leaves earlier than the latest of the jobs in  $\{1, 2, \dots, k-1\}$ . As  $C_k > D_{k-2}$ , it must be that  $C_k > M_{k-2}$ , because  $D_{k-2}$  is the latest departure of the jobs in  $\{1, 2, \dots, k-2\}$ , and this is also equal to  $M_{k-2}$ . As a consequence, if  $C_k < M_{k-1}$ , job  $k$  is also the first job that leaves after  $D_{k-2}$  (provided of course that  $C_{k+1} < C_k$ ). Thus, all in all  $D_{k-1} = \min\{M_{k-1}, C_k\}$ .

In an attempt to extend the above to  $m > 2$  servers, I came up with this scheme:

$$\begin{aligned} A_k &= A_{k-1} + X_k, \\ C_k &= \max\{A_k, D_{k-m}\} + S_k, \\ M_k &= \max\{M_{k-m+1}, C_k\}, \\ D_{k-m+1} &= \min\{M_{k-m+1}, C_k\}, \end{aligned}$$

but it is not correct. Can you find a counterexample?





## FROM TRANSIENT TO STEADY-STATE ANALYSIS

---

With the tools developed in Chapter 1 we can simulate queueing processes. The next step is to develop mathematical models of queueing systems. The aim of this chapter is to make start with this. However, as we will see in Section 2.2, the mathematical analysis of the time-dependent behavior of queueing systems is beyond our capabilities; the transient behavior of even the simplest queueing system is already extremely complicated. Thus, we have to lower our goals, and for this reason we will focus the steady-state behavior of queueing systems. Intuitively speaking, this requires the system to be stable, for otherwise the queue length process grows to infinity.

We introduce concepts of stability and load in Section 2.3 and express these in terms of the arrival, service and departure rates. The notions of arrival and service rate are crucial because they capture our intuition that when jobs arrive faster on average than they can leave, the queue must ‘explode’. As we will see, when the arrival rate is smaller than the service rate, the system is stable. Once stability is ensured, we can properly define in Section 2.4 a number of measures to characterize the performance of the queueing system, such as the average waiting time. In Chapter 3 we will see how to use the load and expected service time to approximate expected waiting times in queue.

Before introducing these definitions, however, we need to introduce some notational short-hands to characterize the type of queueing process, which is the topic of Section 2.1. We provide in Section 2.5 an overview of the relations we introduce in this chapter.

### 2.1 KENDALL’S NOTATION

As became apparent in Sections 1.2 and 1.4, the construction of any single-station queueing process involves three main elements: the distribution of the inter-arrival times between consecutive jobs, the distribution of the service times of the individual jobs, and the number of servers present to process jobs. In this characterization, it is implicit that the inter-arrival times form a set of i.i.d. (independent and identically distributed) random variables, the service times are also i.i.d., and finally, the inter-arrival times and service times are mutually independent.

#### *Theory and Exercises*

To characterize the type of queueing process it is common to use *Kendall’s abbreviation*  $A/B/c/K$ , where  $A$  is the distribution of the inter-arrival times,  $B$  the distribution of the service times,  $c$  the number of servers, and  $K$  the system size, i.e., the total number of customers that can be simultaneously present, whether in queue or in service.<sup>1</sup> In this notation it is assumed that jobs are served in first-in-first-out (FIFO) order; FIFO scheduling is also often called first-come-first-served (FCFS).

<sup>1</sup> The meaning of  $K$  differs among authors. Sometimes it stands for the capacity of the queue, not the entire system. In this book  $K$  corresponds to the system’s size.

When at an arrival a number of jobs arrive simultaneously (like a bus at a restaurant), we say that a batch arrives. Likewise, the server can work in batches, for instance, when an oven processes multiple jobs at the same time. We write  $A^X/B^Y/c$  to denote that  $X$  is the distribution of the arrival batch size and  $Y$  is the distribution of the service batch sizes. When  $X \equiv Y \equiv 1$ , i.e., single batch arrivals and single batch services, we suppress the  $X$  and  $Y$  in the queueing formula.

Two inter-arrival and service distributions are the most important in queueing theory: the exponential distribution denoted with the shorthand  $M$ , as it is memoryless, and a general distribution (with the implicit assumption that its first moment is finite) denoted with  $G$ . We write  $D$  for a deterministic (constant) random variable.

Familiarize yourself with this notation as it is used continuously in the rest of the book. Here are some exercises to illustrate the notation.

**2.1.1.** *What is the meaning of  $M/M/1$ ?*

**2.1.2.** *By how many parameters is the  $M/M/1$  queue characterized?*

**2.1.3.** *What is the  $D/D/1$  queue?*

**2.1.4.** *Is it true that the  $M/M/c$  shorthand means that jobs arrive as a Poisson process, job service times are exponentially distributed, and there are  $c$  servers.*

**2.1.5.** *What is the meaning of  $M/M/c/K$ ?*

**2.1.6.** *What is the meaning of  $M/M/c/c$ ?*

**2.1.7.** *What is the meaning of  $M(n)/M(n)/1$ ?*

**2.1.8.** *What is the meaning of  $M^X/M/1$ ?*

**2.1.9.** *What is the meaning of  $M/G/1$ ?*

**2.1.10.** *What is the meaning of  $M/G/\infty$ ?*

**2.1.11.** *What is the meaning of  $G/G/1$ ?*

**2.1.12.** *What is the meaning of  $M/D/1 - LIFO$ ?*

**2.1.13.** *Is the  $M/D/1$  queue a specific type of  $M/G/c$  queue?*

You should also understand the differences between different scheduling rules. The next exercise should help with this.

**2.1.14.** *What are some advantages and disadvantages of using the Shortest Processing Time First (SPTF) rule to serve jobs?*

When a customer finds a large queue in front of it, s/he can use the normal distribution to estimate the distribution of the time s/he will spend in queue. The next exercise shows how.

**2.1.15.** *Suppose for the  $G/G/1$  that a job sees  $n$  jobs in the system upon arrival. Use the central limit theorem to estimate the distribution of the waiting time in queue for this job.*

**2.1.16.** A machine serves two types of jobs. The processing time of jobs of type  $i$ ,  $i = 1, 2$ , is exponentially distributed with parameter  $\mu_i$ . The type  $T$  of a job is random and independent of anything else, and such that  $P(T = 1) = p = 1 - q = 1 - P(T = 2)$ . (An example is a desk serving men and women, both requiring different average service times, and  $p$  is the probability that the customer in service is a man.) Show that the expected processing time and variance are given by

$$\begin{aligned} E[S] &= p E[S_1] + q E[S_2] \\ V[S] &= p V[S_1] + q V[S_2] + pq(E[S_1] - E[S_2])^2. \end{aligned}$$

Observe that, even if  $V[S_1] = V[S_2] = 0$ , still  $V[S] > 0$  if  $E[S_1] \neq E[S_2]$ . Thus, mixing jobs of constant size still leads to variability in the service process.

Interestingly, we will use these results in also in Section 3.3 when we discuss the effects of failures on average waiting times in the system

#### Hints

**h.2.1.14.** Look up the relevant definitions on Wikipedia or [Hall \[1991\]](#).

**h.2.1.15.** Let  $W_{Q,n} = \sum_{k=1}^n S_k$ . Since  $\{S_k\}$  are assumed to be i.i.d. for the  $G/G/1$  queue,  $W_{Q,n}$  has mean  $\mu_n = n E[S]$  and  $\sigma_n^2 = n V[S]$ .

**h.2.1.16.** Let  $S$  be the processing (or service) time at the server, and  $S_i$  the service time of a type  $i$  job. Then,

$$S = \mathbb{1}_{T=1} S_1 + \mathbb{1}_{T=2} S_2.$$

#### Solutions

**s.2.1.1.**  $M/M/1$ : The distribution of the inter-arrival times is memoryless, hence exponential, the service times are also memoryless, and there is 1 server. As  $K$  is unspecified, the system can contain any number of jobs.

**s.2.1.2.** The inter-arrival times are exponentially distributed with rate  $\lambda$ ; the service times are also exponential, but with parameter  $\mu$ . Thus, if we know  $\lambda$  and  $\mu$ , we have fully characterized the parameters of both distributions. Since the number of servers is 1, only  $\lambda$  and  $\mu$  remain.

**s.2.1.3.**  $D/D/1$ : A queueing process with deterministic inter-arrival times, deterministic service times and 1 server.

**s.2.1.4.**  $M/M/c$ : A multi-server queue with  $c$  servers in which all servers have the same service capacity. Jobs arrive according to a Poisson process and have exponentially distributed service times. Hence, it is true.

**s.2.1.5.**  $M/M/c/K$ : Inter-arrival times and process times are exponential, and the system capacity is  $K$  jobs. Thus, the queue can contain at most  $K - c$  jobs.

**s.2.1.6.**  $M/M/c/c$ : In this system the number of servers is the same as the system capacity, thus the queue length is always zero. This queueing system is useful to determine the number of beds in a hospital; the beds act as servers.

**s.2.1.7.**  $M(n)/M(n)/1$ : The inter-arrival times are exponential, just as the service times, but the rates of the arrival and service processes may depend on the queue length  $n$ .

**s.2.1.8.**  $M^X/M/1$ : Customers arrive with exponentially distributed inter-arrival times. However, each customer brings in a number of jobs, known as a batch. The number of jobs in each batch is distributed as the random variable  $X$ . Thus, the arrival process of work is *compound Poisson*.

**s.2.1.9.**  $M/G/1$ : The inter-arrival times are exponentially distributed, the service times can have any general distribution (with finite mean), and there is 1 server.

**s.2.1.10.**  $M/G/\infty$ : Exponential inter-arrival times, service times can have any distribution, and there is an unlimited supply of servers. This is also known as an *ample server*. Observe that in this queueing process, jobs actually never have to wait in queue; upon arrival there is always a free server available.

**s.2.1.11.**  $G/G/1$ : Generally distributed inter-arrival and service times, 1 server.

**s.2.1.12.**  $M/D/1 - LIFO$ : Now, job service times are deterministic, and the service sequence is last-in-first-out (LIFO).

**s.2.1.13.** Yes, take  $G = D$  and  $c = 1$ .

**s.2.1.14.** Advantage: SPTF minimizes the number of jobs in queue. Thus, if you want to keep the shop floor free of jobs, then this is certainly a good rule. Disadvantage: large jobs get near to terrible waiting times, and the variance of the waiting time increases. Thus, the  $C_s^2$  is larger than under FIFO. Also, SPTF does not take due dates into account, thus giving a reliable due date quotation to a customer is hard (near to impossible).

**s.2.1.15.** Under conditions you can find on the internet,

$$\frac{W_{Q,n} - \mu_n}{\sigma_n} \rightarrow \mathcal{N}(0, 1), \quad \text{as } n \rightarrow \infty,$$

where  $\mathcal{N}(0, 1)$  is a normally distributed random variable with  $\mu = 0$  and  $\sigma^2 = 1$ . But then

$$\begin{aligned} \frac{W_{Q,n} - \mu_n}{\sigma_n} &\approx \mathcal{N}(0, 1) \iff \\ W_{Q,n} - \mu_n &\approx \sigma_n \mathcal{N}(0, 1) \iff \\ W_{Q,n} - \mu_n &\approx \mathcal{N}(0, \sigma_n^2) \iff \\ W_{Q,n} &\approx \mu_n + \mathcal{N}(0, \sigma_n^2) \iff \\ W_{Q,n} &\approx \mathcal{N}(\mu_n, \sigma_n^2) = \mathcal{N}(n \mathbb{E}[X], n \mathbb{V}[S]), \end{aligned}$$

where we used in the last equation the fact that the variance of a (fixed) sum of i.i.d random variables is equal to the sum of the variances.

**s.2.1.16.** With the hint,

$$\begin{aligned} \mathbb{E}[S] &= \mathbb{E}[\mathbb{1}_{T=1}S_1] + \mathbb{E}[\mathbb{1}_{T=2}S_2] \\ &= \mathbb{E}[\mathbb{1}_{T=1}] \mathbb{E}[S_1] + \mathbb{E}[\mathbb{1}_{T=2}] \mathbb{E}[S_2], \text{ by the independence of } T, \end{aligned}$$

$$\begin{aligned}
&= P(T=1) E[S_1] + P(T=2) E[S_2] \\
&= p E[S_1] + q E[S_2].
\end{aligned}$$

For the variance, we need some algebra. Since,

$$\mathbb{1}_{T=1} \mathbb{1}_{T=2} = 0 \text{ and } \mathbb{1}_{T=1}^2 = \mathbb{1}_{T=1},$$

we get

$$\begin{aligned}
V[S] &= E[S^2] - (E[S])^2 \\
&= E[(\mathbb{1}_{T=1}S_1 + \mathbb{1}_{T=2}S_2)^2] - (E[S])^2 \\
&= E[\mathbb{1}_{T=1}S_1^2 + \mathbb{1}_{T=2}S_2^2] - (E[S])^2 \\
&= p E[S_1^2] + q E[S_2^2] - (E[S])^2 \\
&= p V[S_1] + p(E[S_1])^2 + q V[S_2] + q(E[S_2])^2 - (E[S])^2 \\
&= p V[S_1] + p(E[S_1])^2 + q V[S_2] + q(E[S_2])^2 - p^2(E[S_1])^2 - q^2(E[S_2])^2 - 2pq E[S_1] E[S_2] \\
&= p V[S_1] + q V[S_2] + pq E[S_1]^2 + pq E[S_2]^2 - 2pq E[S_1] E[S_2], \quad \text{as } p = 1 - q \\
&= p V[S_1] + q V[S_2] + pq(E[S_1] - E[S_2])^2.
\end{aligned}$$

## 2.2 QUEUEING PROCESSES AS REGULATED RANDOM WALKS

In this section, we provide an elegant construction of a queueing process  $\{L_k\}$  based on a *random walk*  $\{Z_k\}$ . This serves two aims. The first is to show that a characterization of the transient behavior of a queueing process is typically extremely hard. Thus, in the rest of the book we will only study queueing systems in steady-state. The second is to show that queueing theory is essentially based on concepts of fundamental interest in probability theory (the random walk), hence is strongly related to many other applications of random walks, such as in finance, inventory theory, and insurance theory. Moreover, we can use tools of random walk to analyze queueing systems, such as a characterization of the distribution of the time until an especially large queue length is reached; we refer the reader to literature for this.

### Theory and Exercises

In the construction of queueing processes as set out in Section 1.2 we are given two sequences of i.i.d. random variables: the number of arrivals  $\{a_k\}$  per period and the service capacities  $\{c_k\}$ , cf., (1.2.2). Observe that in (1.2.2) the process  $\{L_k\}$  shares a resemblance to a random walk  $\{Z_k, k = 0, 1, \dots\}$  with  $Z_k$  given by

$$Z_k = Z_{k-1} + a_k - c_k. \quad (2.2.1)$$

To see that  $\{Z_k\}$  is indeed a random walk, observe that  $Z$  makes jumps of size  $a_k - c_k, k = 1, \dots$ , and  $\{a_k - c_k\}$  is a sequence of i.i.d. random variables since, by assumption,  $\{a_k\}$  and  $\{c_k\}$  are i.i.d. Clearly,  $\{Z_k\}$  is ‘free’, i.e., it can take positive and negative values, but  $\{L_k\}$  is restricted to the non-negative integers.

#### 2.2.1. Show that $L_k$ satisfies the relation

$$L_k = Z_k - \min_{1 \leq i \leq k} Z_i \wedge 0, \quad (2.2.2)$$

where  $Z_k$  is defined by the above random walk and we write  $a \wedge b$  for  $\min\{a, b\}$ .

This recursion for  $L_k$  leads to really interesting graphs. In Fig. 5 we take  $a_k \sim B(0.3)$ , i.e.,  $a_k$  is Bernoulli distributed with success parameter  $p = 0.3$ , i.e.,  $P(a_k = 1) = 0.3 = 1 - P(a_k = 0)$ , and  $c_k \sim B(0.4)$ . In Fig. 6,  $a_k \sim B(0.49)$  and the random walk is constructed as

$$Z_k = Z_{k-1} + 2a_k - 1. \quad (2.2.3)$$

Thus, if  $a_k = 1$ , the random walk increases by one step, while if  $a_k = 0$ , the random walk decreases by one step, so that  $Z_k \neq Z_{k-1}$  always. Observe that this is slightly different from a random walk that satisfies (2.2.1); there,  $Z_k = Z_{k-1}$ , if  $a_k = c_k$ .

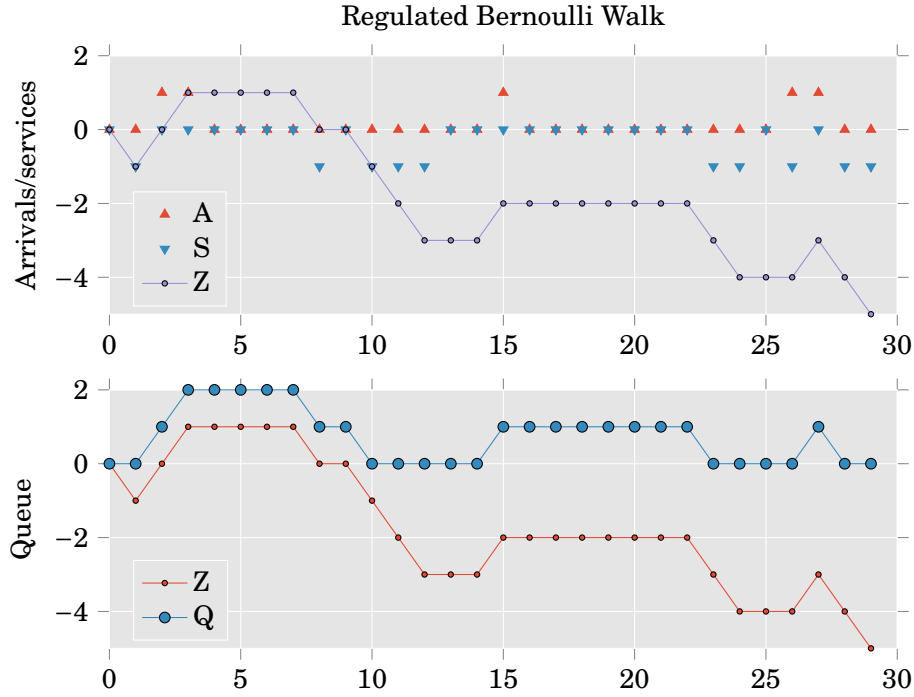


Figure 5: The upper panel shows a graph of the random walk  $Z$ . An upward pointing triangle corresponds to an arrival, a downward triangle to a potential service. The lower panel shows the queueing process  $\{L_k\}$  as a random walk with reflection.

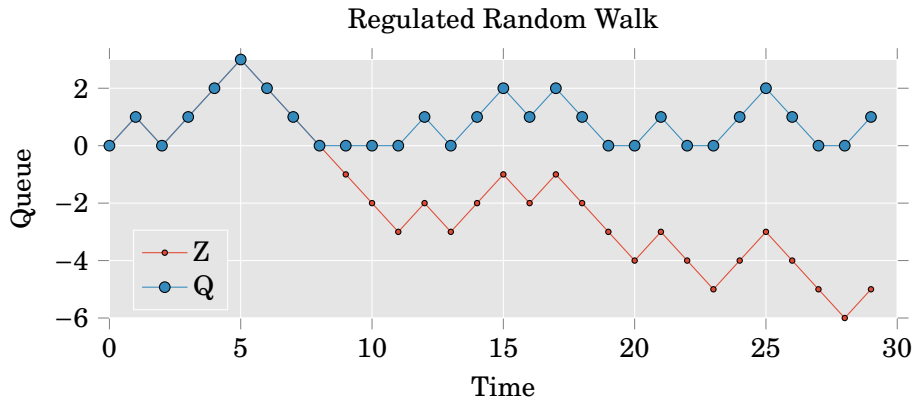


Figure 6: Another example of a reflected random walk.

With (2.2.2), we see that a random walk  $\{Z_k\}$  can be converted into a queueing process  $\{L_k\}$ , and we might try to understand the transient behavior of the latter by investigating the transient behavior of the former. Suppose that  $a_k \sim P(\lambda)$  and  $c_k \sim P(\mu)$ .

**2.2.2.** Show that if  $\{a_k\}$  forms an i.i.d. sequence of random variables all Poisson distributed  $P(\lambda)$  then,  $\sum_{j=1}^k a_j = P(\lambda k)$ .

With the above exercise,

$$Z_k = Z_0 + N_{\lambda k} - N_{\mu k},$$

and we call  $Z = \{Z_k\}$  the *free* (discrete-time)  $M/M/1$  queue as, contrary to the real  $M/M/1$  queue,  $Z$  can take negative values.

**2.2.3.** Show that when  $n > m$  and  $Z_0 = m$ ,

$$P(Z_k = n) = e^{-(\lambda+\mu)k} (\lambda k)^{n-m} \sum_{j=0}^{\infty} \frac{(\lambda \mu k^2)^j}{j!(n-m+j)!}.$$

The solution of the above exercise shows that there is no simple function by which we can compute the transient distribution of this simple random walk  $Z$ . Since a queueing process is typically a more complicated object (as we need to obtain  $L$  from  $Z$  via (2.2.2)), our hopes of finding anything simple for the transient analysis of the  $M/M/1$  queue should not be too high. But the  $M/M/1$  queue is about the simplest queueing system; other queueing systems are (much) more complicated. We therefore give up the analysis of the transient behavior of queueing systems and henceforth contend ourselves with the analysis of queueing systems in the limit as  $t \rightarrow \infty$ .

### Hints

**h.2.2.1.** Note first that from the expression for  $Z_k$ ,  $a_k - c_k = Z_k - Z_{k-1}$ . Use this to get  $L_k = [L_{k-1} + Z_k - Z_{k-1}]^+$ . Subtract  $Z_k$  from both sides, use recursion and use subsequently,

$$\begin{aligned} \max\{\max\{a, b\}, c\} &= \max\{a, b, c\}, \\ L_0 &= Z_0, \\ \max\{-a, -b\} &= -\min\{a, b\}. \end{aligned}$$

**h.2.2.2.** Use [1.1.18].

### Solutions

**s.2.2.1.** Note first that from the expression for  $Z_k$ ,  $a_k - c_k = Z_k - Z_{k-1}$ . Using this in the recursion for  $L_k$ , we get

$$L_k = [L_{k-1} + Z_k - Z_{k-1}]^+,$$

thus,

$$L_k - Z_k = \max\{L_{k-1} - Z_{k-1}, -Z_k\}.$$

From this, using recursion and the hints, we see that

$$\begin{aligned}
 L_k - Z_k &= \max\{\max\{L_{k-2} - Z_{k-2}, -Z_{k-1}\}, -Z_k\} \\
 &= \max\{L_{k-2} - Z_{k-2}, -Z_{k-1}, -Z_k\} \\
 &= \max\{L_0 - Z_0, -Z_1, \dots, -Z_k\} \\
 &= \max\{0, -Z_1, \dots, -Z_k\} \\
 &= -\min\{0, Z_1, \dots, Z_k\}.
 \end{aligned}$$

For further discussion, if you are interested, see [Baccelli and Massey \[1988\]](#).

**s.2.2.2.** Write  $a$  for the common random variable. Then

$$M_a(s) = \mathbb{E}[e^{sa}] = \sum_{j=0}^{\infty} e^{sj} e^{-\lambda j} \lambda^j / j! = e^{\lambda(e^s - 1)}.$$

Then with  $X = \sum_{j=1}^k a_j$ , and by independence,

$$M_X(s) = (M_a(s))^k = e^{\lambda k(e^s - 1)}.$$

Hence,  $X \sim P(\lambda k)$ .

**s.2.2.3.**

$$\begin{aligned}
 P(Z_k = n) &= P(m + N_{\lambda k} - N_{\mu k} = n) \\
 &= P(N_{\lambda k} - N_{\mu k} = n - m) \\
 &= \sum_{j=0}^{\infty} P(N_{\lambda k} = n - m + j, N_{\mu k} = j) \\
 &= \sum_{j=0}^{\infty} e^{-\lambda k} \frac{(\lambda k)^{n-m+j}}{(n-m+j)!} e^{-\mu k} \frac{(\mu k)^j}{j!} \\
 &= e^{-(\lambda+\mu)k} (\lambda k)^{n-m} \sum_{j=0}^{\infty} \frac{(\lambda \mu k^2)^j}{j!(n-m+j)!}.
 \end{aligned} \tag{2.2.4}$$

## 2.3 RATE, STABILITY AND LOAD

In this section, we develop a number of essential concepts to analyze queueing systems: the arrival, service and departure rate. With these we define the load, which is, arguably, the most important performance measure of a queueing system to check.

### Theory and Exercises

We first formalize the arrival rate and departure rate in terms of the arrival and departure processes  $\{A(t)\}$  and  $\{D(t)\}$ ; recall that these are *counting processes*. The *arrival rate* is the long-run average number of jobs that arrive per unit time along a sample path, i.e.,

$$\lambda = \lim_{t \rightarrow \infty} \frac{A(t)}{t}. \tag{2.3.1}$$

We remark in passing that this limit does not necessarily exist if  $A(t)$  is some pathological function. If, however, the inter-arrival times  $\{X_k\}$  are the basic data, and  $\{X_k\}$  are i.i.d. and



distributed as a generic random variable  $X$  with finite mean  $E[X]$ , we can construct  $\{A_k\}$  and  $\{A(t)\}$  as described in Section 1.4, and then the strong law of large numbers guarantees that the above limit exists.<sup>2</sup>

**2.3.1.** *Can you make an arrival process such that  $A(t)/t$  does not have a limit?*

Likewise, define the *departure rate* as

$$\delta = \lim_{t \rightarrow \infty} \frac{D(t)}{t}. \quad (2.3.2)$$

Observe that, if the system is empty at time 0, the number of departures must be smaller than or equal to the number of arrivals, i.e.,  $D(t) \leq A(t)$  for all  $t$ . Therefore,

$$\delta = \lim_{t \rightarrow \infty} \frac{D(t)}{t} \leq \lim_{t \rightarrow \infty} \frac{A(t)}{t} = \lambda. \quad (2.3.3)$$

It is evident that when  $\lambda > \delta$ , the system length process  $L(t) \rightarrow \infty$  as  $t \rightarrow \infty$ . We therefore call a system *rate-stable* if

$$\lambda = \delta.$$

In words: the system is rate-stable whenever jobs leave the system just as fast as they arrive in the long run.

**2.3.2.** *If the system starts empty, then we know that the number  $L(t)$  in the system at time  $t$  is equal to  $A(t) - D(t)$ . Show that the system is rate-stable if  $L(t)$  remains finite, or, more generally,  $L(t)/t \rightarrow 0$  as  $t \rightarrow \infty$ .*

We next relate the arrival rate  $\lambda$  to the expected inter-arrival time  $E[X]$ . Observe that at time  $t = A_n$  precisely  $n$  arrivals occurred. Thus, we see that  $A(A_n) = n$ , and therefore

$$\frac{1}{n} \sum_{k=1}^n X_k = \frac{A_n}{n} = \frac{A_n}{A(A_n)}.$$

But since  $A_n \rightarrow \infty$  if  $n \rightarrow \infty$ , it follows from (2.3.1) that the average inter-arrival time between two consecutive jobs is

$$E[X] = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n X_k = \lim_{n \rightarrow \infty} \frac{A_n}{A(A_n)} = \lim_{t \rightarrow \infty} \frac{t}{A(t)} = \frac{1}{\lambda}, \quad (2.3.4)$$

where we take  $t = A_n$  in the limit for  $t \rightarrow \infty$ . In words, the arrival rate  $\lambda$  is the *inverse* of the expected inter-arrival time  $E[X]$ .

Assume now that there is a single server. Let  $S_k$  be the required service time of the  $k$ th job to be served, so that  $U_n = \sum_{k=1}^n S_k$  becomes the total service time required by the first  $n$  jobs. With this, let  $U(t) = \max\{n : U_n \leq t\}$ , so that we can define the *service rate* or *processing rate* as

$$\mu = \lim_{t \rightarrow \infty} \frac{U(t)}{t}.$$

Similar to the relation  $E[X] = 1/\lambda$ , we have the relation

$$E[S] = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n S_k = \lim_{n \rightarrow \infty} \frac{U_n}{n} = \lim_{n \rightarrow \infty} \frac{U_n}{U(U_n)} = \lim_{t \rightarrow \infty} \frac{t}{U(t)} = \frac{1}{\mu}.$$

<sup>2</sup> In fact,  $A(t)/t \rightarrow \lambda$  with probability one.

Thus, the service rate  $\mu$  is the *inverse* of the expected service time  $E[S]$ .

It turns out that, when  $\mu = \lambda$  and  $V[S] > 0$  or  $V[X] > 0$ , the queue length process behaves in a very peculiar way. This follows from the fact that the random walk without drift, i.e.,  $\mu = \lambda$ , has some unexpected behavior<sup>3</sup>, and we know from Section 2.2 that queueing systems and random walks are intimately related. To avoid such problems, we henceforth (and implicitly) require that  $\mu > \lambda$ . Observe also the evident fact that jobs cannot depart faster than they can be served, hence,  $D(t) \leq U(t)$  for all  $t$ , hence  $\delta \leq \mu$ . Combining this with  $\lambda < \mu$  and rate-stability, we see that  $\delta < \mu$ .

**2.3.3.** Define the random variables  $\{\tilde{X}_k, k = 1, \dots\}$  as  $\tilde{X}_k = S_{k-1} - X_k$ . For stability of the queueing process it is essential that  $\tilde{X}_k$  has negative expectation, i.e.,  $E[\tilde{X}_k] = E[S_{k-1} - X_k] < 0$ . What is the conceptual meaning of this inequality?

**2.3.4.** Define  $\tilde{X}_k = S_{k-1} - X_k$ . Show that  $E[\tilde{X}_k] < 0$  implies that  $\lambda < \mu$ .

Perhaps the most important performance measure is the concept of *load*  $\rho$ , which is defined as the rate at which jobs arrive times the average amount of work per job:  $\rho = \lambda E[S]$ . From the identities  $\lambda^{-1} = E[X]$  and  $\mu^{-1} = E[S]$ , we get some further relations:

$$\rho = \lambda E[S] = \frac{\lambda}{\mu} = \frac{E[S]}{E[X]}. \quad (2.3.5)$$

Observe that by assumption  $\mu > \lambda$ , hence  $\rho < 1$ . The relation  $\rho = E[S]/E[X] < 1$  then tells us that the average time it takes to serve a job must be less than the average time between two consecutive arrivals, i.e.,  $E[S] < E[X]$ . Also, when  $\lambda < \mu$ , it is easy to check with simulation that when  $L(0)$  is very large,  $L(t) \approx L(0) - (\mu - \lambda)t$  until the system is empty, while if  $\lambda > \mu$ , we have that  $L(t) \approx L(0) + (\lambda - \mu)t$ .

**2.3.5.** Consider a queueing system with  $c$  servers with identical production rates  $\mu$ . What would be a reasonable stability criterion for this system?

**2.3.6** (Not obligatory). In (2.3.4) we replaced the limit with respect to  $n$  by a limit with respect to  $t$ . Use the notation  $A_{A(t)}$  to show that this is allowed. Show next that the function  $t \rightarrow A(t)$  as defined by (1.4.3) is right-continuous.

### Hints

**h.2.3.1.** As a start, the function  $\sin(t)$  does not have a limit as  $t \rightarrow \infty$ . However, the time-average  $\sin(t)/t \rightarrow 0$ . Now you need to make some function whose time-average does not converge, hence it should grow fast, or fluctuate wilder and wilder.

**h.2.3.4.** Remember that  $\{X_k\}$  and  $\{S_k\}$  are sequences of i.i.d. random variables. What are the implications for the expectations?

**h.2.3.5.** What is the rate in, and what is the service capacity?

**h.2.3.6.** Use that  $A_{A(t)} \leq t < A_{A(t)+1}$ . Divide by  $A(t)$  and take suitable limits. BTW, such type of proof is used quite often to show that the existence of one limit implies, and is implied by, the existence of another type of limit.

<sup>3</sup> See any good book on probability theory.

### Solutions

**s.2.3.1.** If  $A(t) = 3t^2$ , then clearly  $A(t)/t = 3t$ . This does not converge to a limit.

Another example, let the arrival rate  $\lambda(t)$  be given as follows:

$$\lambda(t) = \begin{cases} 1 & \text{if } 2^{2k} \leq t < 2^{2k+1} \\ 0 & \text{if } 2^{2k+1} \leq t < 2^{2(k+1)}, \end{cases}$$

for  $k = 0, 1, 2, \dots$ . Let  $A(t) = \lambda(t)t$ . Then  $A(t)/t$  does not have a limit. Of course, these examples are quite pathological, and are not representable for ‘real life cases’. (Although this is also quite vague. What, then, is a real-life case?)

For the mathematically interested, we seek a function whose Cesàro limit does not exist.

**s.2.3.2.** Since  $L(t) = A(t) - D(t)$ ,

$$\lambda = \lim_{t \rightarrow \infty} \frac{A(t)}{t} = \lim_{t \rightarrow \infty} \frac{D(t) + L(t)}{t} = \lim_{t \rightarrow \infty} \frac{D(t)}{t} + \lim_{t \rightarrow \infty} \frac{L(t)}{t} = \delta.$$

Hence,  $\lambda = \delta$  when  $L(t)/t \rightarrow 0$ .

**s.2.3.3.** That the average time customers spend in service is smaller than the average time between the arrival of two subsequent jobs.

**s.2.3.4.**  $0 > E[\tilde{X}_k] = E[S_{k-1} - X_k] = E[S_{k-1}] - E[X_k] = E[S] - E[X]$ , where we use the fact that the  $\{S_k\}$  and  $\{X_k\}$  are i.i.d. sequences. Hence,

$$E[X] > E[S] \iff \frac{1}{E[S]} > \frac{1}{E[X]} \iff \mu > \lambda.$$

**s.2.3.5.** The criterion is that  $c$  must be such that  $\lambda < c\mu$ . (Thus, we interpret the number of servers as a *control*, i.e., a ‘thing’ we can change, while we assume that  $\lambda$  and  $\mu$  cannot be easily changed.) To see this, we can take two different points of view. Imagine that the  $c$  servers are replaced by one server that works  $c$  times as fast. The service capacity of these two systems (i.e., the system with  $c$  servers and the system with one fast server) is the same, i.e.,  $c\mu$ , where  $\mu$  is the rate of one server. For the system with the fast server, the load is defined as  $\rho = \lambda/c\mu$ , and for stability we require  $\rho < 1$ . Another way to see it is to assume that the stream of jobs is split into  $c$  smaller streams, each with arrival rate  $\lambda/c$ . In this case, applying the condition that  $(\lambda/c)/\mu < 1$  per server leads to the same condition that  $\lambda/(c\mu) < 1$ .

**s.2.3.6.** This exercise is meant to provide some insight into what needs to be done to put everything on solid ground.

Observing that  $A_{A(t)}$  is the arrival time of the last job before time  $t$  and that  $A_{A(t)+1}$  is the arrival time of the first job after time  $t$ :

$$A_{A(t)} \leq t < A_{A(t)+1} \iff \frac{A_{A(t)}}{A(t)} \leq \frac{t}{A(t)} < \frac{A_{A(t)+1}}{A(t)} = \frac{A_{A(t)+1}}{A(t)+1} \frac{A(t)+1}{A(t)}.$$

Now  $A(t)$  is a counting process such that  $A(t) \rightarrow \infty$  as  $t \rightarrow \infty$ . Therefore,  $\lim_{t \rightarrow \infty} A_{A(t)}/A(t) = \lim_{n \rightarrow \infty} A_n/n$ . Moreover, it is evident that  $\lim_{t \rightarrow \infty} A_{A(t)+1}/(A(t)+1) = \lim_{t \rightarrow \infty} A_{A(t)}/A(t)$ , and that  $(A(t)+1)/A(t) \rightarrow 1$  as  $t \rightarrow \infty$ . Thus it follows from the above inequalities that  $\lim_{n \rightarrow \infty} A_n/n = \lim_{t \rightarrow \infty} t/A(t)$ .

For the right-continuity of  $A(t)$ , define  $f(t) = \mathbb{1}_{A_1 \leq t}$ . Observe first that  $f(t)$  is increasing, and  $f(t) \in \{0, 1\}$ . Thus, if  $f(t) = 1$  then  $f(u) = 1$  for all  $u \geq t$ , and if  $f(t) = 0$  then  $f(u) = 0$  for all  $u \leq t$ .

You may skip the rest of the proof below, but the above is essential to memorize; make a plot of  $f(t)$ , in particular, the behavior around  $A_1$  is important.

We need to prove, for right-continuity, that  $f(u) \rightarrow f(t)$  as  $u \downarrow t$ . When  $f(t) = 1$ ,  $f(u) = 1$  for any  $u > t$ , by the definition of  $f(x)$ . When  $f(t) = 0$  we have to do a bit more work. Formally, we have to prove that, for fixed  $t$  and for all  $\epsilon > 0$ , there is a  $\delta > 0$  such that  $u \in (t, t + \delta) \Rightarrow |f(u) - f(t)| < \epsilon$ . (Note the differences with the regular definition of continuity.) Since, by assumption,  $t$  is such that  $f(t) = 0$ , and  $f \in \{0, 1\}$  we need to show that  $f(u) = 0$  for  $u \in (t, t + \delta)$ . Now, clearly,  $f(t) = 0$  only if  $t < A_1$ . But, then for any  $u \in (t, A_1)$ , we have that  $f(u) = 0$ . Thus, taking  $\delta = A_1 - t$  suffices.

The next step is to observe that  $A(t)$  is a sum of right-continuous functions whose steps do not overlap since by assumption  $0 < A_1 < A_2 < \dots$ . As  $A$  is (almost surely) a finite sum of bounded, increasing and right-continuous functions, it is also right-continuous.

If you like, you can try to prove this last step too.

Hopefully this problem, and its solution, clarifies that even such small details require attention. If we want to make some progress with respect to developing some queueing theory, we have to skip most of the proofs and mathematical problems; we simply don't have enough time in this course to be concerned with all theorems and proofs.

## 2.4 (LIMITS OF) EMPIRICAL PERFORMANCE MEASURES

If the queueing system is rate-stable, we can sensibly define a number of long-run average performance measures such as the load, the average waiting time in queue, and so on. This we do here and refer to Fig. 8 for an overview of the relations between these performance measures.

### Theory and Exercises

Recall that we constructed a single-server queueing process  $\{L(t)\}$  in cf. Section 1.4 based on the arrival process  $\{A(t)\}$  and service process  $\{S(t)\}$ . Once we have the queueing process, we can compute the waiting time process  $\{W_{Q,k}\}$ , the sojourn time process  $\{W_k\}$ .

Define the *expected sojourn time as seen by arrivals* as

$$\mathbb{E}[W] = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n W_k. \quad (2.4.1)$$

Note that this performance measure is the limit of an *empirical* performance measure as observed by arriving jobs: the first job experiences a sojourn time  $W_1$  when it arrives, the second a sojourn time  $W_2$ , and so on. For this reason, we colloquially say such a performance measure is as 'seen by arrivals'. Similarly, the *expected waiting time in queue* is defined as

$$\mathbb{E}[W_Q] = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n W_{Q,k}. \quad (2.4.2)$$

The *distribution of the sojourn times* seen by arrivals can be found by counting:

$$\mathbb{P}(W \leq x) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n \mathbb{1}_{W_k \leq x}. \quad (2.4.3)$$

The (sample) *average number of jobs* in the system as seen by arrivals is given by

$$E[L] = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n L(A_k -), \quad (2.4.4)$$

since  $L(A_k -)$  is the number of jobs in the system at the arrival epoch of the  $k$ th job. Finally, the *distribution of  $\{L(t)\}$*  as seen by arrivals, is given by

$$P(L \leq m) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n \mathbb{1}_{L(A_k -) \leq m}. \quad (2.4.5)$$

A related set of performance measures follows by tracking the system's behavior over time and taking the *time-average*, rather than the average at arrival epochs. Assuming the limit exists we use (1.4.10) to define the *time-average number of jobs* as

$$E[L] = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t L(s) ds. \quad (2.4.6)$$

Observe that, notwithstanding that the symbols are the same, this expectation is not necessarily the same as (2.4.4). In a loose sense we can say that  $E[L]$  is the average number in the system as perceived by the *server*. Next, define the *time-average fraction of time the system contains at most  $m$  jobs* as

$$P(L \leq m) = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t \mathbb{1}_{L(s) \leq m} ds. \quad (2.4.7)$$

Again, this probability need not be the same as what customers see upon arrival.

**2.4.1.** *Design a queueing system to show that the average number of jobs in the system as seen by the server can be very different from what customers see upon arrival.*

**2.4.2.** *If  $L(t)/t \rightarrow 0$  as  $t \rightarrow \infty$ , can it still be true that  $E[L] > 0$ ?*

Taking the above limits requires considerable care, and at least two questions are important. What type of limit do we actually mean here? And, once such limits are given proper meaning, what is the rate of convergence of, for instance, the random variables  $\{L_k\}$  to the limiting random variable  $L$ ? Here we sidestep all such fundamental issues, but see Remark 2.4.1. Assuming that the first of these questions is answered, the limiting random variables are known as the *steady-state or stationary limits*, and the related distributions are often called *limiting or stationary distributions*.

Related to the second question about the convergence rate, we provide some intuition by means of an example. We consider the sequence of waiting times  $\{W_{Q,k}\}$  to a limiting random variable  $W_Q$ , where  $W_{Q,k}$  is constructed according to the recursion (1.4.5). Suppose that  $X_k \sim U\{1, 2, 4\}$  and  $S_k \sim U\{1, 2, 3\}$ . Starting with  $W_{Q,0} = 5$  we use (1.4.5) to compute the *exact* distribution of  $W_{Q,k}$  for  $k = 1, 2, \dots, 20$ , cf., the left panel in Fig. 7. We see that when  $k = 5$ , the ‘hump’ of  $P(W_{Q,5} = x)$  around  $x = 5$  is due the starting value of  $W_{Q,0} = 5$ . However, for  $k > 10$  the distribution of  $W_{Q,k}$  hardly changes, at least not visually. Apparently, the convergence of the sequence of distributions of  $W_{Q,k}$  is rather fast.

In the middle panel we show the results of a set of *simulations* for increasing simulation length, up to  $N = 1000$  samples, where we use the *empirical distribution*

$$P(W_Q \leq x) = \frac{1}{n} \sum_{k=1}^n \mathbb{1}_{W_{Q,k} \leq x}$$

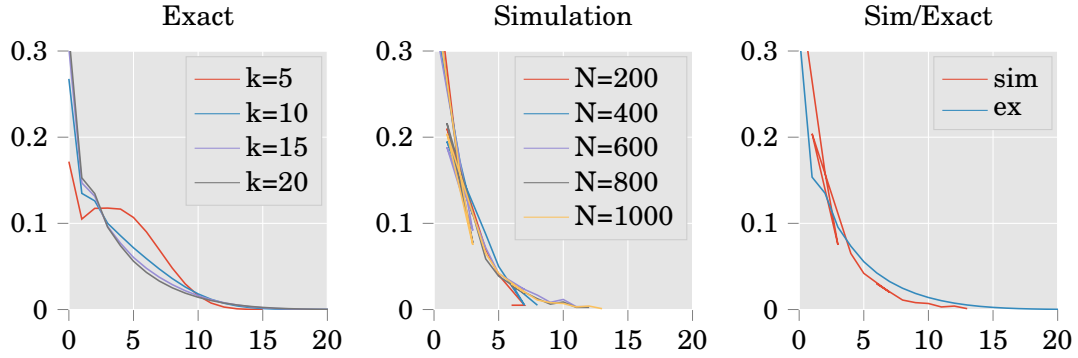


Figure 7: The density of  $W_{Q,k}$  for  $k = 5, 10, 15, 20$  computed by an exact method as compared the density obtained by simulation of different run lengths  $N = 200, 400, \dots, 1000$ . The right panel compares the exact density of  $W_{Q,20}$  to the density obtained by simulation for  $N = 1000$ .

to estimate the waiting time distribution. As should be clear from the figure, the simulated distribution also converges quite fast to some limiting function. Finally, in the right panel we compare the densities as obtained by the exact method and simulation with  $n = 1000$ . Clearly, for all practical purposes, these densities can be treated as the same.

The combination of the fast convergence to the steady-state situation and the difficulties with the transient analysis validates, to some extent, that most queueing theory is concerned with the analysis of the system in *stationary* or *steady state*.

**2.4.3.** Suppose that  $X_k \in \{1, 3\}$  such that  $P(X_k = 1) = P(X_k = 3)$  and  $S_k \in \{1, 2\}$  with  $P(S_k = 1) = P(S_k = 2)$ . Write a computer program to see how fast the distributions of  $W_{Q,k}$  converge to a limiting distribution function.

Up to now, we considered queueing systems in continuous time. However, the performance measures for discrete-time queueing models needs some modification, mainly for the reason that multiple jobs can arrive in a single period. What statistics do these jobs assemble? The next exercise focuses on this problem.

**2.4.4.** Consider the discrete-time model of the queueing system specified by (1.2.1). In such queueing systems, jobs arrive in batches, for instance, when  $a_k = 3$ , three jobs arrive in slot  $k$ . Assuming that  $L_{k-1} = 5$ , what queue length have these 3 arrivals seen?

Provide a definition similar to (2.4.5) for the case in which we say that all arrivals of a batch see the same number in the system. For this, assume that the batch sees, upon arrival,  $L_{k-1} - d_k$  jobs in the system. In other words, we assume that in a period the served jobs first depart before new jobs arrive.

Provide another definition to express that the first of a batch of arrivals sees less jobs in the system than the last arrival of a batch. (To get the details right is harder than you might think; I did it wrong several times.)

**Remark 2.4.1.** The long-run limiting behavior of a queueing system (i.e., the first question) is a subtle topic. For instance, does there exist a random variable  $L$  such that  $L_k \rightarrow L$  in some sense? The answer to this question requires a considerable amount of mathematics. To sketch what has to be done, realize that we first need to define  $\{L_k\}$  as a stochastic process (up to now we just considered each  $L_k$  as a number, i.e., a measurement or simulation of the queue length time of the  $k$ th period.) The construction of  $L_k$  as a proper random variable is not as simple as the

definition of the service times  $\{S_k\}$ , for instance. In the latter case, we just assume these random variables to be i.i.d. However, in the former case, it is apparent from (1.4.7) that the queue lengths  $\{L_k\}$  are constructed in terms of recursions, hence they are certainly not i.i.d. Next, based on these recursions, we need to show that the sequence of distribution functions  $\{G_k\}$  associated with the random variables  $\{L_k\}$  converges to some limiting distribution function  $G$ , say. Finally, it is necessary to show that it is possible to construct a random variable  $L$  that has  $G$  as its distribution function. In this sense, finally, we say that  $L_k \rightarrow L$ . All this can be done, but we refer to the specialized literature for this.

### Hints

**h.2.4.1.** Consider a queueing system with constant service and inter-arrival times.

**h.2.4.4.** Realize that when jobs arrive in batches, the definition of loss fraction requires some care; not all definitions need to measure the same.

### Solutions

**s.2.4.1.** Take  $X_k = 10$  and  $S_k = 10 - \epsilon$  for some tiny  $\epsilon$ . Then  $L(t) = 1$  nearly all of the time. In fact,  $E[L] = 1 - \epsilon/10$ . However,  $L(A_k -) = 0$  for all  $k$ .

**s.2.4.2.**

$$E[L] = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t L(s) ds \neq \lim_{t \rightarrow \infty} \frac{L(t)}{t}.$$

If  $L(t) = 1$  for all  $t$ ,  $E[L] = 1$ , but  $L(t)/t \rightarrow 0$ .

**s.2.4.3.** Here is an example with Python. I compute the difference, i.e., the Kolmogorov-Smirnov statistic, between the distributions of  $W_{Q,k-1}$  and  $W_{Q,k}$ ,

$$\max_x \{ |P(W_{Q,k} \leq x) - P(W_{Q,k-1} \leq x)| \},$$

for  $x$  in the support of  $W_{Q,k}$ .

The code can be found in the exact function in the file `waiting_time_simulation.py` at my github repo.

If you make a plot, you will see that after some 10 customers the distribution hardly changes any further.

**s.2.4.4.** Recall, we assume that arriving jobs in period  $k$  cannot be served in period  $k$ .

Suppose we do not want to distinguish between jobs in a batch; we only count a ‘success’ when all  $a_k$  jobs see  $m$  or less in the system. Since, by assumption, the served jobs first depart before we let new jobs arrive, the first of the  $a_k$  arriving jobs sees a queue length of  $L_{k-1} - d_k$ , the second  $L_{k-1} - d_k + 1$ , and so on, until the last job of the batch, which sees a queue length of  $L_{k-1} - d_k + a_k - 1$ . (Thus, of all  $a_k$  jobs, the last sees the largest queue.) Because we only a success if all jobs of the batch see a ‘small’ queue, we only need to count batches when  $L_{k-1} - d_k + a_k - 1 \leq m$ . But,  $L_{k-1} - d_k + a_k = L_k - 1$ , so that

$$\frac{1}{A(n)} \sum_{k=1}^n a_k \mathbb{1}_{L_k - 1 \leq m},$$

is the fraction we seek.

Below is not ok! I'll repair it.

For the second case, as long as  $j$  is such that  $L_{k-1} - d_k + j - 1 \leq m$ , all these  $j$  jobs fit. Thus, of the  $a_k$  jobs, a number of

$$\min\{a_k, m + 1 - L_{k-1} + d_k\}$$

see at most  $m$  in the system. And therefore, the performance measure becomes

$$\frac{1}{A(n)} \sum_{k=1}^n \max\{a_k, m + 1 - L_{k-1} + d_k\}.$$

Here is the code for the second case.

```
>>> a = [0, 2, 5, 1, 2]
>>> c = [0, 1, 1, 0, 0]

>>> d = [0] * len(a)
>>> L = [0] * len(a)

>>> for k in range(1, len(a)):
...     d[k] = min(L[k - 1], c[k])
...     L[k] = L[k - 1] + a[k] - d[k]
...
>>> print(L)
[0, 2, 6, 7, 9]

>>> m = 5

>>> tot = 0
>>> for k in range(1, len(a)):
...     tot += max(a[k], m+1 - L[k-1] +d[k])
...
>>> print(tot, tot / sum(a))
14 1.4
```

## 2.5 GRAPHICAL SUMMARY

Here is, in graphical form, an overview to show the relation between the concepts developed in this chapter.



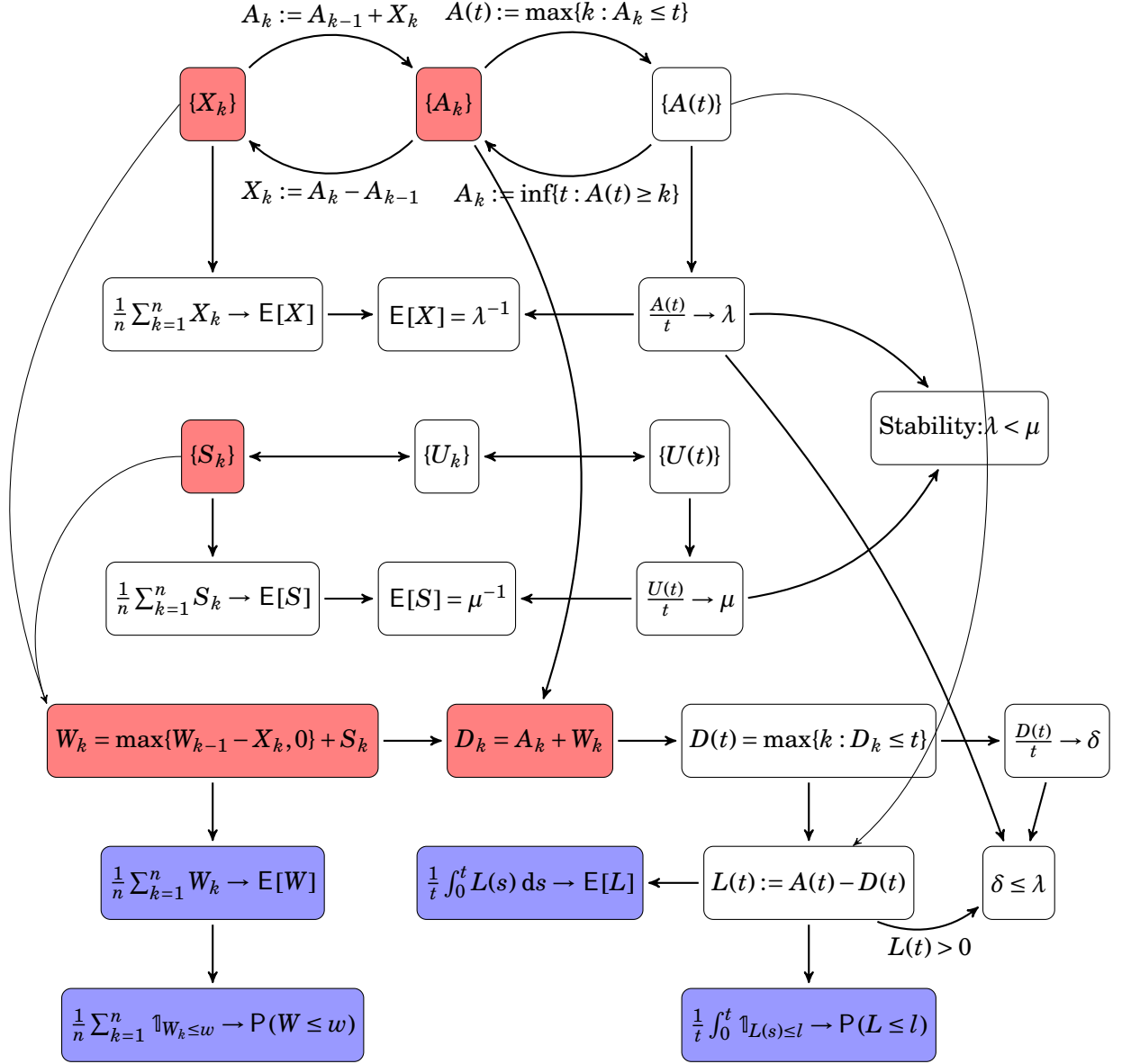


Figure 8: Here we sketch the relations between the construction of the G/G/1 queue from the primary data, i.e., the inter-arrival times  $\{X_k; k \geq 0\}$  and the service times  $\{S_k; k \geq 0\}$ , and different performance measures.



## APPROXIMATE QUEUEING MODELS

---

In this chapter we familiarize the reader with Sakasegawa's formula by which the expected queueing time in a  $G/G/c$  can be approximated, and another formula by which it is possible to characterize how variability propagates through a tandem network of  $G/G/c$  queues. We might say that these two formulas are the most important formulas to understand the behavior of queueing systems. With a bit of exaggeration, it is justified to say that the entire philosophy behind lean manufacturing and the world-famous Toyota production system are based on the principles that can be derived from these two formulas.

Here we take these formulas for granted and focus on the insights they provide and how to use them to guide improvement procedures for practical queueing problems that are, in some way or another, often encountered in production and service systems. In later sections, most notably Section 5.4, we will provide the theoretical background of Sakasegawa's formula.

In Section 3.1 we introduce and discuss the main insights of Sakasegawa's formula. Then, in the subsequent three sections, we illustrate how to use this formula to estimate waiting times in three queueing settings in which the service process is interrupted. In the first case, Section 3.2, the server has to produce jobs from different families, and there is a change-over time required to switch from one production family to another. As such setups reduce the time the server is available to serve jobs, the load must increase. In fact, to reduce the load, the server produces in batches of fixed sizes. In the second case, in Section 3.3, the server sometimes requires small adjustments, for instance, to prevent the production quality to degrade below a certain level. Clearly, such adjustments are typically not required during a job's service; however, they can occur at arbitrary moments in time. Thus, this is different from batch production in which the batch sizes are constant. In the third example, in Section 3.4, quality problems or break downs can occur during a job's service. In the final Section 3.5 we concentrate on tandem queues and study the consequences of two different scenarios to improve a simple network of queues. With the tools we develop here it becomes possible to analyze various scenarios to organize a process and quantitatively compare the scenarios.

In passing we use some interesting results of probability theory and the Poisson process; we will use these again, for instance in Chapter 6.

### 3.1 $G/G/c$ QUEUE: SAKASEGAWA'S FORMULA

In manufacturing settings it is quite often the case that the arrival process at a station is not a Poisson process. For instance, if processing times at a station are nearly constant, and the jobs of this station are sent to a second station for further processing, the inter-arrival times at the second station must be more or less equal too. Hence, in this case, the arrival process at this second station is most probably more regular than at the first station. As a second, trivial, case if the inter-arrival times of jobs are 1 hour always and service times 59 minutes always, there simply cannot be a queue.

In this section, we discuss Sakasegawa's formula by which we can estimate the expected waiting time in queue for the  $G/G/c$  queue. The aim here is to show how to use this formula. In later sections we will provide some theoretical background.

### Theory and Exercises

While there is no closed-form expression available to compute the expected waiting time in queue for the  $G/G/c$  queue, Sakasegawa's formula provides a reasonable approximation. This takes the form

$$E[W_Q] = \frac{C_a^2 + C_s^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} E[S], \quad (3.1.1)$$

where  $\lambda$  is the rate at which jobs arrive at the system,  $E[S]$  is the expected service time,

$$\rho = \frac{\lambda E[S]}{c}$$

is the *load* of the station (but not of the individual machines),  $C_a^2 = V[X]/(E[X])^2$  is the SCV of the inter-arrival times,  $C_s^2 = V[S]/(E[S])^2$  of the service times, and  $c$  the number of servers.

It is crucial to memorize the *scaling* relations that can be obtained from this formula. Even though the above results are only approximations, they prove to be exceedingly useful when designing queueing systems and analyzing the effect of certain changes, in particular changes in processing speed (i.e., service times) and service variability. Roughly:

1.  $E[W_Q] \sim (1-\rho)^{-1}$ . The consequence is that the waiting time increases *very steeply* when  $\rho$  is large. Hence, the waiting time is extremely sensitive to the actual value of  $\rho$  when  $\rho$  is large.
2.  $E[W_Q] \sim C_a^2$  and  $E[W_Q] \sim C_s^2$ . Hence, reductions of the variation of the inter-arrival and service times do affect the waiting time, but only linearly.
3.  $E[W_Q] \sim E[S]/c$ . This means that, from the perspective of a job in queue, the average services are  $c$  times as short.

These insights prove very useful when trying to reduce waiting times in any practical situation. First try to reduce the load (by blocking demand or increasing the capacity), then try to reduce the variability (e.g., by planning the arrival times of jobs), and finally, attempt to split jobs into multiple smaller jobs and use the resulting freedom to reschedule jobs in the queue.

**3.1.1.** Show for the  $M/G/1$  and  $M/M/1$  queue that the approximation (3.1.1) reduces to

$$E[W_Q(M/G/1)] = \frac{1+C_s^2}{2} \frac{\rho}{1-\rho} E[S], \quad E[W_Q(M/M/1)] = \frac{\rho}{1-\rho} E[S].$$

The above results explain part of the form of Sakasegawa's formula. In case  $C_a^2 = C_s^2 = 1$ , we obtain the expression for  $E[W_Q(M/M/1)]$ . As we will see in Section 5.1, this is an exact result. Then, by generalizing to the  $M/G/1$  queue, we obtain  $(1+C_s^2)/2$

Section 4.4, and Section 5.4, these results are exact.

**3.1.2.** (Hall 5.19) When a bus reaches the end of its line, it undergoes a series of inspections. The entire inspection takes 5 minutes on average, with a standard deviation of 2 minutes. Buses

arrive with inter-arrival times uniformly distributed on  $[3, 9]$  minutes, hence, 10 buses arrive on average per hour.

As a first case, assuming a single server, estimate  $E[W_Q]$  with the G/G/1 waiting time formula. As a second case, compare this result to an M/G/1 system with arrival rate 10 per hour and the same service time distribution.

Explain why the queueing time in the first setting is smaller.

The next exercise shows us that in a queue with finite capacity, there is a relation between the loss and the average number of servers occupied. Thus, if we want to reduce the loss, we need to reduce the load on the servers.

**3.1.3.** Consider a queue with  $c$  servers, with generally distributed inter-arrival times, generally distributed service times, and the system can contain at most  $K$  customers, i.e., the G/G/c/K queue. Let  $\lambda$  be the arrival rate,  $\mu$  the service rate,  $\beta$  the long-run fraction of customers lost, and  $\rho$  the average number of busy/occupied servers. Show that

$$\beta = 1 - \rho \frac{\mu}{\lambda}.$$

Clearly, Sakasegawa's equation requires an estimate of  $C_a^2$  and  $C_s^2$ . Now it is not always easy in practice to determine the actual service time distribution, one reason being that service times are often only estimated by a planner, but not actually measured. Similarly, the actual arrival moments of jobs are often not registered, only just the date, or perhaps the hour, that a customer arrived. Hence, it is often not possible to estimate  $C_a^2$  and  $C_s^2$  from the information that is available. However, when for instance the number of arrivals per period has been logged for some time so that we know the arrivals per period  $\{a_n, n = 1, \dots, N\}$  for some  $N$ , we can use this information instead of the inter-arrival times  $\{X_k\}$  to obtain insight into  $C_a^2$ . The relation we present here to compute  $C_a^2$  from  $\{a_n\}$  can of course also be applied to estimate  $C_s^2$ .

**Theorem 3.1.1.** The SCV of the inter-arrival times can be estimated with the formula

$$C_a^2 \approx \frac{\tilde{\sigma}^2}{\tilde{\lambda}},$$

where

$$\tilde{\lambda} = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n a_i, \quad \tilde{\sigma}^2 = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n a_i^2 - \tilde{\lambda}^2.$$

In words,  $\tilde{\lambda}$  is the average number of arrivals per period, e.g., per day, and  $\tilde{\sigma}^2$  is the variance of the number of arrivals per period.

*Proof.* The proof is based on an argument in Cox [1962]. We use quite a bit of the notation developed in Section 2.3. Let  $\{A(t), t \geq 0\}$  be the number of arrivals that occur up to (and including) time  $t$ . We assume that  $\{A(t)\}$  is a renewal process such that the inter-arrival times  $\{X_k, k = 1, 2, \dots\}$  with  $X_k = A_k - A_{k-1}$ , are i.i.d. with mean  $1/\lambda$  and standard deviation  $\sigma$ . (Observe that  $\sigma$  is not the same as  $\tilde{\sigma}$  above.) Note that  $C_a^2$  is defined in terms of  $\lambda$  and  $\sigma$  as:

$$C_a^2 = \frac{V[X_i]}{(E[X_i])^2} = \frac{\sigma^2}{1/\lambda^2} = \lambda^2 \sigma^2.$$

Next, let  $A_k$  be the arrival time of the  $k$ th arrival. The following useful relation between  $A(t)$  and  $A_k$  enables us to prove our result (recall that we used a similar relation in the derivation of the Poisson process):

$$P(A(t) < k) = P(A_k > t).$$

Since the inter-arrival times have finite mean and second moment by assumption, we can apply the central limit law to obtain that, as  $k \rightarrow \infty$ ,

$$\frac{A_k - k/\lambda}{\sigma\sqrt{k}} \rightarrow N(0, 1),$$

where  $N(0, 1)$  is a standard normal random variable with distribution  $\Phi(\cdot)$ . Similarly,

$$\frac{A(t) - \lambda t}{\alpha\sqrt{t}} \rightarrow N(0, 1)$$

for an  $\alpha$  that is yet to be determined. Thus,  $E[A(t)] = \lambda t$  and  $V[A(t)] = \alpha^2 t$ .

Using that  $P(N(0, 1) \leq y) = P(N(0, 1) > -y)$  (and  $P(N(0, 1) = y) = 0$ ) we have that

$$\begin{aligned} \Phi(y) &\approx P\left(\frac{A_k - k/\lambda}{\sigma\sqrt{k}} \leq y\right) \\ &= P\left(\frac{A_k - k/\lambda}{\sigma\sqrt{k}} > -y\right) \\ &= P\left(A_k > \frac{k}{\lambda} - y\sigma\sqrt{k}\right). \end{aligned}$$

Define for ease

$$t_k = \frac{k}{\lambda} - y\sigma\sqrt{k}.$$

We can use the above relation between the distributions of  $A(t)$  and  $A_k$  to see that  $P(A_k > t_k) = P(A(t_k) < k)$ . With this we get,

$$\begin{aligned} \Phi(y) &\approx P(A_k > t_k) \\ &= P(A(t_k) < k) \\ &= P\left(\frac{A(t_k) - \lambda t_k}{\alpha\sqrt{t_k}} < \frac{k - \lambda t_k}{\alpha\sqrt{t_k}}\right). \end{aligned}$$

Since  $(A(t_k) - \lambda t_k)/\alpha\sqrt{t_k} \rightarrow N(0, 1)$  as  $t_k \rightarrow \infty$ , the above implies that

$$\frac{k - \lambda t_k}{\alpha\sqrt{t_k}} \rightarrow y,$$

as  $t_k \rightarrow \infty$ . Using the above definition of  $t_k$ , the left-hand side of this equation can be written as

$$\frac{k - \lambda t_k}{\alpha\sqrt{t_k}} = \frac{\lambda\sigma\sqrt{k}}{\alpha\sqrt{k/\lambda + \sigma\sqrt{k}}}y.$$

Since  $t_k \rightarrow \infty$  is implied by (and implies)  $k \rightarrow \infty$ , we therefore want that  $\alpha$  is such that

$$\frac{\lambda\sigma\sqrt{k}}{\alpha\sqrt{k/\lambda + \sigma\sqrt{k}}}y \rightarrow y,$$

as  $k \rightarrow \infty$ . This is precisely the case when

$$\alpha = \lambda^{3/2}\sigma.$$

Finally, for  $t$  large (or, by the same token  $k$  large),

$$\frac{\sigma_k^2}{\lambda_k} = \frac{V[A(t)]}{E[A(t)]} \approx \frac{\alpha^2 t}{\lambda t} = \frac{\alpha^2}{\lambda} = \frac{\lambda^3 \sigma^2}{\lambda} = \lambda^2 \sigma^2 = C_a^2,$$

where the last equation follows from the above definition of  $C_a^2$ . The proof is complete.  $\square$

*Hints**Solutions*

**s.3.1.1.** For the  $M/G/1$  queue,  $C_a^2 = 1$ . Then take  $c = 1$  in Sakasegawa's formula. Next, realize that  $C_s^2 = 1$  for the  $M/M/1$  queue.

**s.3.1.2.** First the  $G/G/1$  case. Observe that in this case, the inter-arrival time  $X \sim U[3, 9]$ , that is, never smaller than 3 minutes, and never longer than 9 minutes.

```
>>> a = 3.
>>> b = 9.
>>> EX = (b+a)/2. # expected inter-arrival time
>>> EX
6.0
>>> labda = 1./EX # per minute
>>> labda
0.16666666666666666
>>> VA = (b-a)*(b-a)/12.
>>> CA2 = VA/(EX*EX)
>>> CA2
0.08333333333333333

>>> ES = 5.
>>> sigma = 2
>>> VS = sigma*sigma
>>> CS2 = VS/(ES*ES)
>>> CS2
0.16

>>> rho = labda*ES
>>> rho
0.8333333333333333

>>> Wq = (CA2+CS2)/2. * rho/(1.-rho) * ES
>>> Wq
3.0416666666666665
```

Now the  $M/G/1$  case.

```
>>> Wq = (1.+CS2)/2. * rho/(1.-rho) * ES
>>> Wq
14.499999999999999
```

The arrival process with uniform inter-arrival times is much more regular than a Poisson process. In the first case, bus arrivals are spaced in time at least with 3 minutes.

**s.3.1.3.** This follows from the observation that  $\lambda(1 - \beta)$  is the net arrival rate, as jobs are lost at a rate  $\lambda\beta$ . Hence, the load must be  $\lambda(1 - \beta)/\mu$ . As the load must be equal to  $\rho$ , it follows that  $\rho = \lambda(1 - \beta)/\mu$ , from which the other result immediately follows.

### 3.2 SETUPS AND BATCH PROCESSING

In some cases machines have to be setup before they can start producing items. Consider, for instance, a machine that paints red and blue bikes. When the machine requires a color change, it may be necessary to clean up the machine. Another example is an oven that needs a temperature change when different item types require different production temperatures. Service operations form another setting with setup times: when servers (personnel) have to move from one part of a building to another, the time spent moving cannot be spent on serving customers. In all such cases, the setups consume a significant amount of time; in fact, setup times of an hour or longer are not uncommon. Clearly, in such situations, it is necessary to produce in batches: a server processes a batch of jobs (or customers) of one type or at one location, then the server changes from type or location, starts serving a batch of another type or at another location. Once done with one type, the server is setup again, and so on. Here we focus on the effect of change-over, or setup, times on the average sojourn time of jobs.

First we make a model and provide a list of elements required to compute the expected sojourn time of an item, then we illustrate how to use these elements in a concrete case.

#### *Theory and Exercises*

Specifically, we analyze the following batch queueing situation. There are two job families, e.g., red and blue, each served by the same single server. Jobs arrive at rate  $\lambda_r$  and  $\lambda_b$ , respectively, so that the arrival rate of jobs is  $\lambda = \lambda_b + \lambda_r$ . Jobs of both types require an average *net processing time* of  $E[S_0]$ , provided the server is already setup for the correct job color. The setup times  $\{R_i\}$  are assumed to form an i.i.d. sequence with common random variable  $R$  and independent of  $S_0$ . The sojourn time comprises the following steps. First, jobs of each color are assembled into batches of size  $B$ , which we assume to be the same for both colors. Once a batch is complete, the batch enters a queue (of batches). After some time the batch reaches the head of the queue. Then the machine performs a setup, and starts processing each job individually until the batch is complete. Finally, once a job is finished, it can leave the system; as a consequence, it does not have to wait for other jobs in the same batch to form a new batch.

**3.2.1.** *Show that the average time a job has to wait to fill the batch (to which this job belongs) is given by*

$$E[W_r] = \frac{B-1}{2\lambda_r}. \quad (3.2.1)$$

Now that we have a batch of jobs, we need to estimate the average time a batch spends in queue. For this we can use the  $G/G/1$  waiting time formula, but we have to convert the effects of the setup times into job service times. Define, to this end, the *effective processing time*  $E[S]$  as the average time the server is occupied with processing a job from a batch including the setup time required to setup the batch.

**3.2.2.** *Motivate that the effective processing time of an item should be defined as*

$$E[S] = E[S_0] + \frac{E[R]}{B}. \quad (3.2.2)$$

With the previous exercise, the load becomes

$$\rho = \lambda \left( E[S_0] + \frac{E[R]}{B} \right).$$



There is another important way to look at the load, see the next exercise.

**3.2.3.** Explain that the load can also be written as

$$\rho = \lambda_B (B E[S_0] + E[R]),$$

where  $\lambda_B = \lambda/B$  is the arrival rate of batches and  $E[S_B] = B E[S_0] + E[R]$  is the service time an entire batch.

**3.2.4.** Show that the requirement  $\rho < 1$  leads to the following constraint on the minimal batch size  $B$

$$B > \frac{\lambda E[R]}{1 - \lambda E[S_0]}.$$

Observe that we nearly have all elements to apply Sakasegawa's formula to the batch: the service time of a batch is

$$E[S_B] = E[R] + B E[S_0], \quad (3.2.3)$$

and the load  $\rho$  is given above. Therefore it remains to find  $C_{a,B}^2$  and  $C_{s,B}^2$ , for the batches, not the items.

**3.2.5.** Explain that the SCV of the batch inter-arrival times is given by

$$C_{a,B}^2 = \frac{C_a^2}{B}. \quad (3.2.4)$$

**3.2.6.** Show that the SCV  $C_{s,B}^2$  of the service times of the batches takes the form

$$C_{s,B}^2 = \frac{B V[S_0] + V[R]}{(B E[S_0] + E[R])^2}. \quad (3.2.5)$$

Observe that we now have all elements to fill in Sakasegawa's formula, which becomes

$$E[W_{Q,B}] = \frac{C_{a,B}^2 + C_{s,B}^2}{2} \frac{\rho}{1 - \rho} E[S_B].$$

It is left to find a rule to determine what happens to an item once the batch to which the item belongs enters service. If the job has to wait until all jobs in the batch are served, the expected time an item spends at the server is  $E[R] + B E[S_0]$ .

**3.2.7.** Show that, when items can leave right after being served, the time at the server is given by

$$E[R] + \frac{B+1}{2} E[S_0]. \quad (3.2.6)$$

**3.2.8.** Jobs arrive at  $\lambda = 3$  per hour at a machine with  $C_a^2 = 1$ ; service times are exponential with an average of 15 minutes. Assume  $\lambda_r = 0.5$  per hour, hence  $\lambda_b = 3 - 0.5 = 2.5$  per hour. Between any two batches, the machine requires a cleanup of 2 hours, with a standard deviation of 1 hour, during which it is unavailable for service. What is the smallest batch size that can be allowed?

What is the average time a red job spends in the system in case  $B = 30$  jobs? Finally, observe that there is  $B$  that minimizes the average sojourn time.

**3.2.9.** What important insights can you learn from the above about setting proper batch sizes?

A much more interesting and realistic problem is to assume that we have many families such that items of family  $i$  arrive at rate  $\lambda_i$  and require service time  $S_i$ . Typically, the setup time depends on the sequence in which the families are produced; let this be given by  $R_{ij}$  when the machine switched from producing family  $i$  to family  $j$ . Often, in practice,  $R_{ij} \neq R_{ji}$ , for example, a switch in color from white to black takes less cleaning time and cost than from black to white. Then the problem becomes to determine a good schedule in which to produce the families and the batch size  $B_i$ , which may be depend on the family. Here is an exercise to show how to handle a simple case.

**3.2.10.** Consider a paint factory that contains a paint mixing machine that serves two classes of jobs, A and B. The processing times of jobs of types A and B are constant and require  $t_A$  and  $t_B$  hours. The job arrival rate is  $\lambda_A$  for type A and  $\lambda_B$  for type B jobs. It takes a setup time of  $S$  hours to clean the mixing station when changing from paint type A to type B, and there is no time required to change from type B to A.

To keep the system stable, it is necessary to produce the jobs in batches, for otherwise the server, i.e., the mixing machine, spends a too large fraction of time on setups. Motivate that the following linear program can be used to determine the minimal batch sizes:

minimize  $T$

such that  $T = k_A t_A + S + k_B t_B$ ,  $\lambda_A T < k_A$  and  $\lambda_B T < k_B$ .

*Hints*

**h.3.2.1.** Show that the total time to form a red batch is  $(B - 1)/\lambda_r$ .

**h.3.2.2.** What fraction of the setup time  $E[R]$  ‘belongs’ to one job?

**h.3.2.5.** Use (0.2.5) and (0.2.6).

**h.3.2.6.** What is the variance of a batch service time?

**h.3.2.10.** Here are some questions to help you interpret this formulation.

1. What are the decision variables for this problem? In other words, what are the ‘things’ we can control/change?
2. What are the interpretations of  $k_A t_A$ , and  $S + k_B t_B$ ?
3. What is the meaning of the first constraint? Realize that  $T$  represents one production cycle. After the completion of one such cycle, we start another cycle. Hence, the start of every cycle can be seen as a restart of the entire system.
4. What is the meaning of the other two constraints?
5. Why do we minimize the cycle time  $T$ ?
6. Solve for  $k_A$  and  $k_B$  in terms of  $S$ ,  $\lambda_A$ ,  $\lambda_B$  and  $t_A$ ,  $t_B$ .
7. Generalize this to  $m$  job classes and such that the cleaning time between jobs of class  $i$  and  $j$  is given by  $S_{ij}$ . (Thus, the setup times are sequence-dependent.)

### Solutions

**s.3.2.1.** Suppose a batch is just finished. The first job of a new batch needs to wait, on average,  $B - 1$  inter-arrival times until the batch is complete, the second  $B - 2$  inter-arrival times, and so on. The last job does not have to wait at all. Thus, the total time to form a batch is  $(B - 1)/\lambda_r$ .

An arbitrary job can be anywhere in the batch, hence the average time a job must wait until the batch is complete is half the total time.

**s.3.2.2.** The total service time spent on a batch of size  $B$  is  $B E[S_0] + E[R]$ . The effective time per job is then the average, i.e.,  $(B E[S_0] + E[R])/B$ . Simplifying gives the answer.

**s.3.2.3.** It is evident that the rate at which batches arrive is

$$\lambda_B = \frac{\lambda}{B},$$

since both job colors have the same batch size. Then the equality has the interpretation of the batch arrival rate times the work per batch.

**s.3.2.4.** We require that the load is less than one, in other words, that  $\lambda(E[S_0] + E[R]/B) < 1$ . This is equivalent to  $E[R]/B < 1/\lambda - E[S_0]$ , hence  $B > E[R]/(1/\lambda - E[S_0])$ . Multiplying with  $\lambda$  gives the result.

**s.3.2.5.** The variance of the inter-arrival time of batches is  $B$  times the variance of job inter-arrival times. The inter-arrival times of batches is also  $B$  times the inter-arrival times of jobs. Thus,

$$C_{a,B}^2 = \frac{B V[X]}{(B E[X])^2} = \frac{V[X]}{(E[X])^2} \frac{1}{B} = \frac{C_a^2}{B}.$$

**s.3.2.6.** The variance of a batch is  $V[\sum_{i=1}^B S_{0,i} + R] = B V[S_0] + V[R]$ , since the normal service times  $S_{0,i}, i = 1, \dots, B$  of the job are independent, and also independent of the setup time  $R$  of the batch.

**s.3.2.7.** First, wait until the setup is finished, then wait (on average) for half of the batch (minus the job itself) to be served, and then the job has to be served itself, that is,  $E[R] + \frac{B-1}{2} E[S_0] + E[S_0]$ .

**s.3.2.8.** First check the load.

```
>>> labda = 3 # per hour
>>> ES0 = 15./60 # hour
>>> ES0
0.25
>>> ER = 2.
>>> B = 30
>>> ESe = ES0+ ER/B
>>> ESe
0.31666666666666665

>>> rho = labda*ESe
>>> rho
0.95
```

Evidently, the load is smaller than 1.

The minimal batch size is

```
>>> Bmin = labda*ER/(1-labda*ES0)
>>> Bmin
24.0
```

So, with  $B = 30$  we are on the safe side.

The time to form a red batch is

```
>>> labda_r = 0.5
>>> EW_r = (B-1)/(2*labda_r)
>>> EW_r # in hours
29.0
```

And the time to form a blue batch is

```
>>> labda_b = labda-labda_r
>>> EW_b = (B-1)/(2*labda_b)
>>> EW_b # in hours
5.8
```

Now the time a batch spends in queue

```
>>> Cae = 1.
>>> CaB = Cae/B
>>> CaB
0.03333333333333333
>>> Ce = 1 # SCV of service times
>>> VS0 = Ce*ES0*ES0
>>> VS0
0.0625
>>> VR = 1*1. # Var setups is sigma squared
>>> VSe = B*VS0 + VR
>>> VSe
2.875
>>> ESb = B*ES0+ER
>>> ESb
9.5
>>> CeB = VSe/(ESb*ESb)
>>> CeB
0.03185595567867036
>>> EWq = (CaB+CeB)/2 * rho/(1-rho) * ESb
>>> EWq
5.88333333333333275
```

The time to unpack the batch, i.e., the time at the server.

```
>>> ES = ER + (B-1)/2 * ES0 + ES0
>>> ES
5.875
```

The overall time red jobs spend in the system.

```
>>> total = EW_r + EWq + ES
>>> total
40.758333333333326
```

**s.3.2.9.** We can derive a number of important insights from this model. First, the time to assemble and unpack batches increases linearly of  $B$ . Second, observe from (3.2.4) and (3.2.5) that the SCVs are  $B$  times as small, but in (3.2.3) we see that the batch service time is roughly  $B$  times larger. Thus, the factor  $B$  approximately cancels out in Sakasegawa's formula. However, the load decreases as a function of  $B$ , and this is the most important reason to use batch production. Third, since the load decreases as a function of the  $B$ , the expected time in queue must also decrease. Combining this with the linear batch forming times, there must be a batch size that minimizes the total time in the system. Fourth, with a bit of numerical experimentation will show that the minimum in the waiting time is rather insensitive to  $B$ . However, by making  $B$  smaller, the load increases, and we know that queueing times increase very steeply when the load increases. Therefore, by making  $B$  small, we 'gain' time linearly in batch forming times, but we 'pay' hyperbolically in  $B$ . Given the randomness in practical situations, it is better, when tuning the batch size to minimize the time in the system, to set  $B$  a bit too large than too small.

**s.3.2.10.** Realize that the machine works in cycles. A cycle starts with processing  $k_A$  jobs of type A, then does a setup, and processes  $k_B$  jobs of type B, and then a new cycle starts again. The time it takes to complete one such cycle is  $T = k_A t_A + S + k_B t_B$ . The number of jobs of type A processed during one such cycle is, of course,  $k_A$ . Observe next that the average number of jobs that arrive during one cycle is  $\lambda_A T$ . We of course want that  $\lambda_A T < k_A$ , i.e., fewer jobs of type A arrive on average per cycle than what we can process.

### 3.3 NON-PREEMPTIVE INTERRUPTIONS, SERVER ADJUSTMENTS

In Section 3.2 we studied the effect of setup times between job batches with a fixed size  $B$ . However, other types of interruptions can occur, such as a machine requiring random adjustments that can occur between any two jobs. This type of outages is *non-preemptive* as the outages do not interrupt the processing of a job in service. In this section we develop a simple model to understand the impact of such outages on job sojourn times; we use the same notation as in Section 3.2 and follow the same line of reasoning. With this model, we can analyze the effects of reducing adjustment times or the variability of these adjustments times. For instance, we might decide to do fewer adjustments, but the average outage times become longer.

#### *Theory and Exercises*

We assume that adjustments  $\{R_i\}$  occur geometrically distributed between any two jobs with a mean of  $B$  jobs between any two adjustments. Consequently, the probability of an outage between any two jobs is  $p = 1/B$ . Observe that geometrically distributed random variables satisfy the memoryless property in discrete time. Hence, our assumption implies that the occurrence of an adjustment between jobs  $i$  and  $i + 1$  has no effect on the probability that an adjustment is necessary between jobs  $i + 1$  and  $i + 2$ .

Contrary to the batch processing case of Section 3.2, we now only need to find the mean and variance of the effective processing times.

**3.3.1.** Show that the average effective processing time is given by

$$E[S] = E[S_0] + \frac{E[R]}{B} \quad (3.3.1)$$

Clearly, the effective server load including down-times is  $\rho = \lambda E[S]$ . Since the adjustments do not affect the job arrival process, we only have to find an expression how it affects  $C_s^2$ .

**3.3.2.** Show that

$$E[S^2] = E[S_0^2] + 2 \frac{E[S_0] E[R]}{B} + \frac{E[R^2]}{B}.$$

**3.3.3.** Derive

$$V[S] = V[S_0] + \frac{V[R]}{B} + (B-1) \left( \frac{E[R]}{B} \right)^2. \quad (3.3.2)$$

With this, we have all elements to fill in the  $G/G/1$  waiting time formula!

**3.3.4.** Jobs arrive as a Poisson process with rate  $\lambda = 9$  per working day. The machine works two 8 hour shifts a day. Work not processed on a day is carried over to the next day. Job service times are 1.5 hours, on average, with standard deviation of 0.5 hours. Outages occur on average between 30 jobs. The average duration of an outage is 5 hours and has a standard deviation of 2 hours.

Compute the average waiting time in queue.

**Remark 3.3.1.** Let us compare the results of this section to those of Section 3.3. In both cases the expected service time is the same: an amount  $E[R]/B$  gets added to the net processing time  $E[S_0]$ . However, the impact on the SCV is different. Unexpected interruptions must have a larger effect on variability than expected interruptions.

In more general terms, we can use part of the model of Section 3.2 to analyze the effect of planned maintenance—do an adjustment after every  $B$  jobs—while with the model of this section we consider adjustments that arrive completely unexpectedly. With these two models we can compare the influence on waiting times of doing planned adjustments more often (take  $B$  a bit small), but remove the unexpected interruptions. Hence, we have another set of tools to compare different scenarios to organize production systems.

*Hints*

**h.3.3.1.** If there is no interruption, the service time is  $S_0$ . However, if there is an interruption, the service of a job is  $R + S_0$ , since we add the outage time to the service time of the job. Relate this insight to [2.1.16]. In that exercise, take  $p = (B-1)/B$  as the probability of a short job of size  $E[S_0]$  and  $q = 1/B$  as the probability of a long job of  $E[S_0] + E[R]$ .

**h.3.3.2.** Again, use [2.1.16].

**h.3.3.3.** First compute  $E[S^2]$ . See [3.3.2].

**h.3.3.4.** Get the units right. First compute the load, and then compute the rest.

*Solutions***s.3.3.1.**

$$E[S] = (1-p)E[S_0] + p(E[R] + E[S_0]) = E[S_0] \frac{B-1}{B} + (E[S_0] + E[R]) \frac{1}{B},$$

since  $p = 1/B$ .

**s.3.3.2.**

$$\begin{aligned} E[S^2] &= (1-p)E[S_0^2] + pE[(S_0 + R)^2] \\ &= (1-p)E[S_0^2] + pE[S_0^2] + 2pE[S_0]E[R] + pE[R^2]. \end{aligned}$$

Simplify and substitute  $p = 1/B$ .

**s.3.3.3.**

$$\begin{aligned} V[S] &= E[S^2] - (E[S])^2 \\ &= E[S_0^2] + 2E[S_0]E[R] \frac{1}{B} + E[R^2] \frac{1}{B} \\ &\quad - (E[S_0])^2 - 2E[S_0]E[R] \frac{1}{B} - (E[R])^2 \frac{1}{B^2} \\ &= V[S_0] + (E[R^2] - (E[R])^2) \frac{1}{B} + (E[R])^2 \left( \frac{1}{B} - \frac{1}{B^2} \right). \end{aligned}$$

**s.3.3.4.** First we determine the load.

```
>>> B=30
>>> ES0 = 1.5
>>> labda = 9./(2*8) # arrival rate per hour
>>> ER=5.
>>> ESe=ES0+ER/B
>>> ESe
1.6666666666666667
>>> rho = labda*ESe
>>> rho
0.9375
```

So, at least the system is stable.

```
>>> VS0 = 0.5*0.5
>>> VR = 2.*2.
>>> VSe = VS0 + VR/B + (B-1)*(ER/B)**2
>>> VSe
1.1888888888888887
>>> Ce2 = VSe/(ESe*ESe)
>>> Ce2
0.4279999999999999
```

And now we can fill in the waiting time formula.

```
>>> Ca2=1 # Poisson arrivals
>>> EW = (Ca2+Ce2)/2 * rho/(1-rho) * ESe
>>> EW
17.849999999999998
```

## 3.4 PREEMPTIVE INTERRUPTIONS, SERVER FAILURES

In Sections 3.2 and 3.3 we assumed that servers are never interrupted while serving a job. However, in many situations this assumption is not satisfied: a person might receive a short phone call while working on a job, a machine may fail in the midst of processing, and so on. In this section, we develop a model to compute the influence on the mean waiting time of such *preemptive outages*, i.e., interruptions that occur *during* a service. As in Section 3.3, we can use Sakasegawa's formula to estimate expected queueing time for the  $G/G/1$  queue. For this, it suffices to find expressions for  $E[S]$  and  $V[S]$ .

We will first present the main formulas and show how to apply them. The exercises at the end of the section ask you to provide the derivations. Specifically, here we compute the expectation and variance of a random number of failures, each with a random duration. Similar problems occur in, for instance, insurance theory (or inventory theory) where we are interested in (the distribution of) the total claim (demand) size that occurs in a period. Noting that the total claim (demand) is given by a random number of claims (demands), each of which is a random variable by itself, that problem is exactly the same as the problem here.

The techniques we develop here are of general interest, and they provide relations between the topics discussed in Section 0.2, Section 1.1, Section 5.6, and Section 6.2.

*Theory and Exercises*

As in the previous sections, we derive expressions for the expectation and variance of the effective processing time  $S$ .

Supposing that  $N$  interruptions occur during the net service time  $S_0$  of a job, and the repair times  $\{R_i\}$  form an i.i.d. sequence distributed as a common random variable  $R$ , write

$$S_N = \sum_{i=1}^N R_i \quad (3.4.1)$$

for the total duration of the interruptions that occur during the service of a job. Then, the effective service time becomes

$$S = S_0 + S_N = S_0 + \sum_{i=1}^N R_i.$$

It is important to realize that  $N$  is a random number.

A common assumption is that the time between two interruptions is memoryless; let the interruptions occur at a given *failure rate*  $\lambda_f$ . Then, if the net service time  $S_0$  is a constant, the expected number of failures  $E[N]$  is  $\lambda_f S_0$ . More generally, we show below that  $E[N] = \lambda_f E[S_0]$ .

Define the *availability* as

$$A = \frac{m_f}{m_f + m_r},$$

where  $m_f$  is the mean time to fail and  $m_r$  the mean time to repair. From [3.4.4] it follows that

$$A = \frac{1}{1 + \lambda_f E[R]}. \quad (3.4.2)$$

Then, from [3.4.8],

$$E[S] = \frac{E[S_0]}{A}. \quad (3.4.3)$$



Now the load is easy to compute:

$$\rho = \lambda E[S] = \lambda \frac{E[S_0]}{A}.$$

Clearly, the server load increases due to failures.

By assuming that repair times are exponentially distributed with mean  $E[R]$ , we can find with the exercises below that

$$C_s^2 = C_0^2 + 2A(1-A) \frac{E[R]}{E[S_0]}, \quad (3.4.4)$$

where  $C_0^2$  is the SCV of  $S_0$ , i.e., the service time without interruptions.

Before deriving the above expressions, let us see how to apply them.

**3.4.1.** Suppose we have a machine with memoryless failure behavior, with a mean-time-to-fail of 3 hours. Regular service times are deterministic with an average of 10 minutes, jobs arrive as a Poisson process with rate of 4 per hour. Repair times are exponential with a mean duration of 30 minutes. What is the average sojourn time?

**3.4.2.** Suppose we could buy another machine that never fails. What is the average sojourn time?

The rest of the section is concerned with the derivations of the formulas.

**3.4.3.** Provide an intuitive explanation for  $E[S] = E[S_0]/A$ .

**3.4.4.** Derive (3.4.2) for our model of interruptions.

**3.4.5.** Suppose that the number of failures  $N = n$ , show that  $E[S_n] = n E[R]$ , where  $S_n$  is given by (3.4.1).

**3.4.6.** Use [3.4.5] to show that  $E[S_N] = E[R] E[N]$ . This result is known as Wald's equation.

**3.4.7.** Show that  $E[N] = \lambda_f E[S_0]$  and  $E[N^2] = \lambda_f^2 E[S_0^2] + \lambda_f E[S_0]$ .

**3.4.8.** Show that  $E[S] = E[S_0] + \lambda_r E[S_0] E[R]$ , and conclude that  $E[S] = E[S_0]/A$ .

**3.4.9.** The derivation of  $C_s^2$  is a bit more involved. To understand why, explain first that  $V[S] \neq V[S_0] + V[\sum_{i=0}^N R_i]$ .

**3.4.10.** Show that

$$E[S^2] = E[S_0^2] + 2E\left[S_0 \sum_{i=1}^N R_i\right] + E\left[\sum_{i=1}^N R_i^2\right] + E\left[\sum_{i=1}^N \sum_{j \neq i} R_i R_j\right].$$

**3.4.11.** Show that  $E[S_0 \sum_{i=1}^N R_i] = \lambda_f E[R] E[S_0^2]$ .

**3.4.12.** Show that  $E[\sum_{i=1}^N R_i^2] = \lambda_f E[S_0] E[R^2]$ .

**3.4.13.** Show that  $E[\sum_{i=1}^N \sum_{j \neq i} R_i R_j] = \lambda_f^2 E[S_0^2] (E[R])^2$ .

Principally we are done now. With  $E[S^2]$  and  $E[S]$  we can compute the SCV via  $(E[S^2] - (E[S])^2)/(E[S])^2$ , so that we have all components for Sakasegawa's formula. The rest is polishing.

**3.4.14.** Combine the above to see that

$$E[S^2] = \frac{E[S_0^2]}{A^2} + \lambda_f E[R^2] E[S_0].$$

**3.4.15.** Show that

$$V[S] = \frac{V[S_0]}{A^2} + \lambda_f E[R^2] E[S_0].$$

**3.4.16.** Show that

$$C_s^2 = \frac{V[S]}{(E[S])^2} = C_0^2 + \frac{\lambda_f E[R^2] A^2}{E[S_0]},$$

**3.4.17.** With the above assumption that  $R$  is exponentially distributed, show that

$$C_s^2 = C_0^2 + 2A(1-A) \frac{E[R]}{E[S_0]}.$$

*Hints*

**h.3.4.1.** Mind to work in a consistent set of units, e.g., hours. It is easy to make mistakes.

**h.3.4.4.** Observe that  $m_f = 1/\lambda_f$  and  $m_r = E[R]$ .

**h.3.4.5.** Is it relevant for the expectation of  $S_n$  that  $R_1, \dots, R_n$  are mutually independent?

**h.3.4.6.** Use (0.2.2).

**h.3.4.7.** Use that the joint distribution of  $S_0 = s, N = k$  is given by  $g(s)e^{-\lambda s}(\lambda s)^k/k!$ , where we assume for the sake of simplicity that  $S_0$  has the density  $g$ . Then use [1.1.9].

**h.3.4.8.** Realize that  $E[N] = \lambda_f E[S_0]$ . Then use [3.4.4].

**h.3.4.11.** Use (0.2.2) and [3.4.7].

**h.3.4.12.** Use Wald's equation, which we derived in [3.4.6].

**h.3.4.16.** Just realize that  $E[S] = E[S_0]/A$ , and use the above.

*Solutions*

**s.3.4.1.** Let's first compute the load. If  $\rho > 1$  we are in trouble.

```
>>> labda = 4.
>>> ES0 = 10./60 # in hours
>>> labda_f = 1./3
>>> ER = 30./60 # in hours
>>> A = 1./(1+labda_f*ER)
>>> A
0.8571428571428571
>>> ES = ES0/A
>>> ES
0.19444444444444445
>>> rho = labda*ES
>>> rho
0.7777777777777778
```

As  $\rho < 1$ , the system is not in overload. Now for the queueing time.

```
>>> Ca2 = 1.
>>> C02 = 0. # deterministic service times
>>> Ce2 = C02 + 2*A*(1-A)*ER/ES0
>>> Ce2
0.7346938775510207
>>> EW = (Ca2+Ce2)/2 * rho/(1-rho) * ES
>>> EW
0.5902777777777779
>>> EW + ES # sojourn time
0.7847222222222223
```

**s.3.4.2.** Now we don't need to take availability into account: the machine never fails so  $A = 1$ .

```
>>> labda = 4.
>>> ES0 = 10./60 # in hours
>>> A = 1
>>> ES = ES0/A
>>> rho = labda*ES
>>> rho
0.6666666666666666
>>> Ca2 = 1.
>>> C02 = 0. # deterministic service times
>>> EW = (Ca2+C02)/2 * rho/(1.-rho) * ES
>>> EW
0.16666666666666663
>>> EW + ES # sojourn time
0.33333333333333326
```

The average time in queue reduces from  $\approx 0.6$  to  $\approx 0.17$  hours, a reduction by about a factor 3.

**s.3.4.3.** An intuitive way to obtain this result is by noting that  $A$  is the fraction of time the server is working. As the total service time of a job is  $E[S]$ , the net work done is  $A E[S]$ . But this must be the time needed to do the real job, hence  $A E[S] = E[S_0]$ .

**s.3.4.4.** The time to fail is the time in between two interruptions. By assumption, the failure times are  $\text{Exp}(\lambda_f)$ , hence  $m_f = 1/\lambda_f$ . The expected duration of an interruption is  $E[R] = m_r$ . With this

$$A = \frac{m_f}{m_f + m_r} = \frac{1/\lambda_f}{1/\lambda_f + E[R]}.$$

**s.3.4.5.** The expectation of the *fixed* sum of random variables is the same as the sum of the expectations; independence is irrelevant. Hence,

$$E[S_n] = E\left[\sum_{i=1}^n R_i\right] = n E[R],$$

since by assumption  $\{R_i\}$  have the same distribution, hence  $E[R_i] = E[R]$ .

**s.3.4.6.** Using (0.2.2) and the fact that  $E[S_n] = n E[R]$ , we get

$$\begin{aligned} E[S_N] &= E \left[ \sum_{n=0}^{\infty} \mathbb{1}_{N=n} S_n \right] = E \left[ \sum_{n=0}^{\infty} \mathbb{1}_{N=n} \left( \sum_{i=1}^n R_i \right) \right] \\ &= \sum_{n=0}^{\infty} E[\mathbb{1}_{N=n} n E[R]] \\ &= E[R] \sum_{n=0}^{\infty} n E[\mathbb{1}_{N=n}] = E[R] \sum_{n=0}^{\infty} n p_n \\ &= E[R] E[N], \end{aligned}$$

where we specifically use (0.2.2b) in the second to last equation.

**s.3.4.7.** If  $S = s$ , then, by (1.1.1), the expected number of failures that arrive is  $E[N(s)] = \lambda_f s$ . Therefore,  $E[N] = E[\lambda_f S_0] = \lambda_f E[S_0]$ .

In more detail, and with the hint,

$$E[N] = \int_0^{\infty} \sum_{k=0}^{\infty} k e^{-\lambda_f s} \frac{(\lambda_f s)^k}{k!} g(s) ds = \int_0^{\infty} \lambda_f s g(s) ds = \lambda_f E[S_0].$$

Next, [1.1.9],

$$E[N^2] = \int_0^{\infty} \sum_{k=0}^{\infty} k^2 e^{-\lambda_f s} \frac{(\lambda_f s)^k}{k!} g(s) ds = \int_0^{\infty} (\lambda_f^2 s^2 + \lambda_f s) g(s) ds.$$

**s.3.4.8.**

$$\begin{aligned} E[S] &= E[S_0 + S_N] = E[S_0] + E[S_N] \\ &= E[S_0] + E[N] E[R] = E[S_0] + \lambda_f E[S_0] E[R] = E[S_0] (1 + \lambda_f E[R]). \end{aligned}$$

**s.3.4.9.** Observe that  $S_0$  and  $N$  are not independent. In fact, when  $S_0 = s$ , the number of failures  $N$  is Poisson distributed with mean  $\lambda_f s$ .

**s.3.4.10.** Just work out the square of  $S_0 + \sum_{i=1}^N R_i$  and take expectations. Realize that  $(\sum_i R_i)^2 = \sum_i R_i^2 + \sum_i \sum_{j \neq i} R_i R_j$ .

**s.3.4.11.** With the hint,

$$\begin{aligned} E \left[ S_0 \sum_{i=1}^N R_i \right] &= E \left[ S_0 \sum_{n=0}^{\infty} \mathbb{1}_{N=n} \sum_{i=1}^n R_i \right] \\ &= E \left[ S_0 \sum_{n=0}^{\infty} \mathbb{1}_{N=n} \sum_{i=1}^n R_i \right] \\ &= E[R] E \left[ S_0 \sum_{n=0}^{\infty} \mathbb{1}_{N=n} n \right] \\ &= E[R] \sum_{n=0}^{\infty} n E[S_0 \mathbb{1}_{N=n}]. \end{aligned}$$

Now observe that  $S_0$  and  $\mathbb{1}_{N=n}$  are not independent. Thus, we need to use the joint distribution of [3.4.7], and then,

$$E[S_0 \mathbb{1}_{N=n}] = \int_0^{\infty} s g(s) e^{-\lambda_f s} \frac{(\lambda_f s)^n}{n!} ds.$$

Continuing with the previous relation:

$$\begin{aligned}
 \mathbb{E}[R] \sum_{n=0}^{\infty} n \mathbb{E}[S_0 \mathbb{1}_{N=n}] &= \mathbb{E}[R] \sum_{n=0}^{\infty} n \int_0^{\infty} s g(s) e^{-\lambda_f s} \frac{(\lambda_f s)^n}{n!} ds \\
 &= \mathbb{E}[R] \int_0^{\infty} s g(s) \sum_{n=0}^{\infty} n e^{-\lambda_f s} \frac{(\lambda_f s)^n}{n!} ds \\
 &= \mathbb{E}[R] \int_0^{\infty} s g(s) \lambda_f s ds \\
 &= \lambda_f \mathbb{E}[R] \mathbb{E}[S_0^2].
 \end{aligned}$$

**s.3.4.12.** With the hint,

$$\mathbb{E} \left[ \sum_{i=1}^N R_i^2 \right] = \mathbb{E}[R^2] \mathbb{E}[N] = \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2].$$

**s.3.4.13.** Since the  $\{R_i\}$  are i.i.d.,

$$\begin{aligned}
 \mathbb{E} \left[ \sum_{i=1}^N \sum_{j \neq i} R_i R_j \right] &= \mathbb{E} \left[ \sum_{n=0}^{\infty} \mathbb{1}_{N=n} \sum_{i=1}^N \sum_{j \neq i} R_i R_j \right] \\
 &= \mathbb{E} \left[ \sum_{n=0}^{\infty} \mathbb{1}_{N=n} \sum_{i=1}^n \sum_{j \neq i} R_i R_j \right] \\
 &= \mathbb{E} \left[ \sum_{n=0}^{\infty} \mathbb{1}_{N=n} n(n-1) \right] (\mathbb{E}[R])^2 \\
 &= (\mathbb{E}[N^2] - \mathbb{E}[N]) (\mathbb{E}[R])^2 \\
 &= (\lambda_f^2 \mathbb{E}[S_0^2] + \lambda_f \mathbb{E}[S_0] - \lambda \mathbb{E}[S_0]) (\mathbb{E}[R])^2.
 \end{aligned}$$

where we use [3.4.7] in the last line.

**s.3.4.14.** It is just algebra based on the results above. Observe that  $(1/A) = 1 + \lambda_f \mathbb{E}[R]$ ,

$$\begin{aligned}
 \mathbb{E}[S^2] &= \mathbb{E}[S_0^2] + 2 \mathbb{E} \left[ S_0 \sum_{i=1}^N R_i \right] + \mathbb{E} \left[ \sum_{i=1}^N R_i^2 \right] + \mathbb{E} \left[ \sum_{i=1}^N \sum_{j \neq i} R_i R_j \right] \\
 &= \mathbb{E}[S_0^2] + 2 \lambda_f \mathbb{E}[R] \mathbb{E}[S_0^2] + \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2] + \lambda_f^2 \mathbb{E}[S_0^2] (\mathbb{E}[R])^2 \\
 &= \mathbb{E}[S_0^2]/A + \lambda_f \mathbb{E}[R] \mathbb{E}[S_0^2] + \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2] + \lambda_f^2 \mathbb{E}[S_0^2] (\mathbb{E}[R])^2 \\
 &= \mathbb{E}[S_0^2]/A + \lambda_f \mathbb{E}[R] \mathbb{E}[S_0^2] (1 + \lambda_f \mathbb{E}[R]) + \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2] \\
 &= \mathbb{E}[S_0^2]/A + \lambda_f \mathbb{E}[R] \mathbb{E}[S_0^2]/A + \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2] \\
 &= (1 + \lambda_f \mathbb{E}[R]) \mathbb{E}[S_0^2]/A + \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2] \\
 &= \mathbb{E}[S_0^2]/A^2 + \lambda_f \mathbb{E}[S_0] \mathbb{E}[R^2].
 \end{aligned}$$

**s.3.4.15.**

$$V[S] = \mathbb{E}[S^2] - (\mathbb{E}[S])^2 = \frac{\mathbb{E}[S_0^2]}{A^2} + \lambda_f \mathbb{E}[R^2] \mathbb{E}[S_0] - \frac{(\mathbb{E}[S_0])^2}{A^2}.$$

**s.3.4.16.** Using [3.4.10]–[3.4.15]

$$C_s^2 = \frac{V[S]}{(\mathbb{E}[S])^2} = \frac{V(S)A^2}{(\mathbb{E}[S_0])^2}$$

$$\begin{aligned}
&= \frac{E[S_0^2] + \lambda_f E[R^2] E[S_0] A^2 - (E[S_0])^2}{(E[S_0])^2} \\
&= \frac{E[S_0^2] - (E[S_0])^2}{(E[S_0])^2} + \frac{\lambda_f E[R^2] E[S_0] A^2}{(E[S_0])^2} \\
&= C_0^2 + \frac{\lambda_f E[R^2] A^2}{E[S_0]}.
\end{aligned}$$

**s.3.4.17.** When repair times are exponentially distributed with mean  $E[R]$  we have, by [1.3.3], that  $E[R^2] = 2(E[R])^2$ . Since  $A = 1/(1 + \lambda_f E[R])$ ,

$$\begin{aligned}
\lambda_f E[R^2] A^2 &= 2\lambda_f (E[R])^2 A^2 \\
&= 2 \frac{\lambda_f E[R]}{1 + \lambda_f E[R]} A E[R] \\
&= 2 \left( 1 - \frac{1}{1 + \lambda_f E[R]} \right) A E[R] = 2(1 - A) A E[R].
\end{aligned}$$

### 3.5 TANDEM QUEUES

Consider two  $G/G/1$  stations in tandem. Suppose we have the financial means to reduce the variability of the processing times at one of the servers, but not at both. Then we like to improve the one that has the most impact on the total waiting time in the system. To obtain insight into this problem, we present a formula that approximates the SCV of the inter-departure times of a  $G/G/c$  queue. Noting that the output of the first machine forms the input of the second machine, the SCV of the inter-departure times of first station must then be the SCV of the inter-arrival times at the second station. Thus, with this formula for the SCV of the inter-departure times, we can model the propagation of variability through a simple network of  $G/G/c$  stations in tandem.

We remark that the literature contains algorithms that can deal with more complicated networks of  $G/G/c$  queues in which the output streams of several stations merge into the input stream of another station and rework is allowed. However, when the machines act like  $M/M/c$  queues, it is possible to fully analyze the system, cf. Section 6.3.

#### *Theory and Exercises*

With simulation it has been tested that the SCV of the inter-departure times of a  $G/G/c$  queue can be reasonable well approximated by

$$C_d^2 \approx 1 + (1 - \rho^2)(C_a^2 - 1) + \frac{\rho^2}{\sqrt{c}}(C_s^2 - 1). \quad (3.5.1)$$

**3.5.1.** Show that (3.5.1) reduces to the following for the  $G/G/1$  queue:

$$C_d^2 = (1 - \rho^2)C_a^2 + \rho^2 C_s^2. \quad (3.5.2)$$

Clearly, for single-server queues the expression becomes quite simple for which we provide the following intuitive interpretation. Suppose that the load  $\rho$  is very high. Then the server will seldom be idle, so that most of inter-departure times are the same as service times. If, however, the load is low, the server will be idle most of the time, and the inter-departure times are approximately equal to the inter-arrival times. The approximation then consists of an interpolation between these two extremes.

**3.5.2.** What is  $C_d^2$  for the  $D/D/1$  queue according to (3.5.2)?

**3.5.3.** Show that  $C_d^2 = 1$  for the  $M/M/1$  queue according to (3.5.2). In fact, in [6.3.8] we provide a proof that this is an exact result.

The next exercise proves a very useful insight: if we make service times more regular, the departure process also becomes more regular.

**3.5.4.** Use (3.5.2) to show for the  $G/D/1$  that  $C_d^2 < C_a^2$ .

**3.5.5.** Consider two  $G/G/1$  stations in tandem. Suppose  $\lambda = 2$  per hour,  $C_{a,1}^2 = 2$  at station 1,  $C_s^2 = 0.5$  at both stations, and  $E[S_1] = 20$  minutes and  $E[S_2] = 25$  minutes. What is the total time jobs spend on average in the system?

Let us next apply the above to analyze the performance of a tandem network of two  $M/M/1$  queues. For this, assume that jobs arrive at the first station at rate  $\lambda$ , and are served at rate  $\mu_i$  at station  $i$ . Thus,  $\rho_i = \lambda/\mu_i$  and  $E[S_i] = 1/\mu_i$ , for  $i = 1, 2$ . First we consider the base case, i.e., the system without any improvement at either of the servers. Then we assume we can remove all variability at the second station; this is the best that we can possibly do at station 2. Next, we assume we can remove all variability at the first station. The final step is to compare the different improvement scenarios.

To analyze this case, observe that the departures of the first station form the arrivals at the second station. Thus, the SCV of the inter-arrival times at the second station is the SCV of the inter-departure times of the first station, that is  $C_{d,1}^2 = C_{a,2}^2$ . This idea can of course be used for longer tandem networks.

**3.5.6.** Use Sakagawa's formula to show that the average queueing time for the tandem of two  $M/M/1$  queues is given by

$$E[W_Q] = \frac{\rho_1}{1 - \rho_1} \frac{1}{\mu_1} + \frac{\rho_2}{1 - \rho_2} \frac{1}{\mu_2}, \quad (3.5.3)$$

Now suppose we can remove all variability of the service process at the second station.

**3.5.7.** Show that in this case the total time in queue is equal to

$$E[W_Q] = \frac{\rho_1}{1 - \rho_1} \frac{1}{\mu_1} + \frac{1}{2} \frac{\rho_2}{1 - \rho_2} \frac{1}{\mu_2}.$$

Suppose now that we reduce the variability of the service process of the first station.

**3.5.8.** Motivate that

$$E[W_Q] = \frac{1}{2} \frac{\rho_1}{1 - \rho_1} \frac{1}{\mu_1} + \frac{2 - \rho_1^2}{2} \frac{\rho_2}{1 - \rho_2} \frac{1}{\mu_2}$$

is a reasonable approximation of the queueing time in the network.

**3.5.9.** Comparing these three scenarios, what do you conclude?

*Hints*

**h.3.5.7.** The second station becomes an  $M/D/1$  queue if  $C_{s,2}^2 = 0$ .

**h.3.5.9.** What would you do if there would be a third station in this tandem network?

*Solutions*

**s.3.5.1.** Since  $c = 1$  for the  $G/G/c$  queue, we get

$$\begin{aligned} C_d^2 &= 1 + (1 - \rho^2)(C_a^2 - 1) + \rho^2(C_s^2 - 1) \\ &= 1 + C_a^2 - \rho^2 C_a^2 - 1 + \rho^2 + \rho^2 C_s^2 - \rho^2 \\ &= (1 - \rho^2)C_a^2 + \rho^2 C_s^2. \end{aligned}$$

**s.3.5.2.** Since the inter-arrival times and the service times are deterministic,  $C_a^2 = C_s^2 = 0$ . Hence  $C_d^2 = 0$ .

**s.3.5.3.** Since the inter-arrival times and the service times are exponential,  $C_a^2 = C_s^2 = 1$ . Hence  $C_d^2 = 1$ .

**s.3.5.4.** As  $C_s^2 = 0$  for the  $G/D/1$  queue,  $C_d^2 = (1 - \rho^2)C_a^2 < C_a^2$ , as  $\rho < 1$ .

**s.3.5.5.** First station 1.

```
>>> labda = 2.
>>> S1 = 20./60
>>> rho1 = labda*S1
>>> rho1
0.6666666666666666
>>> ca1 = 2.
>>> cs1 = 0.5
>>> EW1 = (ca1+cs1)/2 * rho1/(1-rho1) * S1
>>> EW1
0.8333333333333331
>>> W1 = EW1 + S1
>>> W1
1.1666666666666665
```

Now station 2. We first need to compute  $C_{d1}^2$ .

```
>>> cd1 = (1-rho1**2)*ca1 + rho1**2*cs1
>>> cd1
1.3333333333333335

>>> labda = 2
>>> S2 = 25./60
>>> rho2 = labda*S2
>>> rho2
0.8333333333333334
>>> ca2 = cd1 # here we use our formula
>>> cs2 = 0.5
>>> EW2 = (ca2+cs2)/2 * rho2/(1-rho2) * S2
>>> EW2
1.9097222222222223
>>> W2 = EW2 + S2
>>> W2
2.3263888888888897
```



**s.3.5.6.** It follows right away from [3.1.1].

**s.3.5.7.** Fill in Sakagawa's formula for the second station with  $C_{s,2}^2 = 0$ .

**s.3.5.8.** When  $C_{s,1}^2 = 0$ , the first station becomes an  $M/D/1$  queue. By (3.5.2),  $C_{d,1}^2 = (1 - \rho_1^2)C_{a,1}^2 + \rho_1^2 C_{s,1}^2 = 1 - \rho_1^2$ , since by assumption  $C_{a,1}^2 = 1$  and  $C_{s,1}^2 = 0$ . Then, at the second station,  $C_{a,2}^2 + C_{s,2}^2 = 1 - \rho_1^2 + 1$ .

**s.3.5.9.** As a general guideline, it seems best to reduce the variability at the first station. The main point to remember is that reducing the variability of the service process at the first station also reduces the variability of its departure process, hence the variability of the arrival processes at the second station becomes also smaller, and so on. In other words, improving the first station can improve things at the entire chain.



## FUNDAMENTAL TOOLS

---

To develop mathematical models of queueing systems we need a few concepts that are fundamentally important and have a general interest beyond queueing. All of these concepts rely on *sample-path constructions* of queueing, or more general stochastic, systems. Thus, sample paths, which are nothing but the output of a simulation, form an elegantly unifying principle.

Here we keep the discussion in these notes mostly at an intuitive level; we refer to [El-Taha and Stidham Jr. \[1998\]](#) for proofs and further background.

### 4.1 RENEWAL REWARD THEOREM

We state the *renewal reward theorem* and provide graphical motivation for its validity. This results proves to be extremely useful: we will apply it regularly in the sequel of the book and here we use it to provide another definition of the load  $\rho$ .

#### *Theory and Exercises*

In essence the renewal reward theorem is very simple: it states that when customers arrive at rate  $\lambda$  and each customer pays an average amount  $X$ , the system earns money at rate  $Y = \lambda X$ . Figure 9 provides graphical motivation about why this theorem is true; [El-Taha and Stidham Jr. \[1998\]](#) gives a (simple) proof.

**Theorem 4.1.1** (Renewal Reward Theorem,  $Y = \lambda X$ ). *Consider epochs  $\{T_k, k = 0, 1, \dots\}$  such that  $0 = T_0 < T_1 < \dots$ . Let  $N = \{N(t), t \geq 0\}$  be the associated counting process with  $N(t) = \max\{k : T_k \leq t\}$ . Let  $\{Y(t), t \geq 0\}$  be a non-decreasing right-continuous (deterministic) process. Define  $X_k = Y(T_k) - Y(T_{k-1})$ . Suppose that  $N(t)/t \rightarrow \lambda$  as  $t \rightarrow \infty$ , where  $0 < \lambda < \infty$ . Then  $Y(t)/t$  has a limit iff  $n^{-1} \sum_{k=1}^n X_k$  has a limit, and then  $Y = \lambda X$ . In other words,*

$$\lim_{t \rightarrow \infty} \frac{Y(t)}{t} = Y \iff \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n X_k = X,$$

and then  $Y = \lambda X$ .

Let us use this theorem to understand the idea of ‘load’ in a different way. Define the *load* or *utilization* as the limiting fraction of time the server is busy, i.e.,

$$\rho = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t \mathbb{1}_{L(s) > 0} ds.$$

**4.1.1.** Use the renewal reward theorem to prove that  $\rho = \lambda E[S]$  for the rate-stable G/G/1 queue.

**4.1.2.** We can derive the relation  $\rho = \lambda E[S]$  in a somewhat more direct way by considering the fact that

$$\sum_{k=1}^{A(t)} S_k \geq \int_0^t \mathbb{1}_{L(s) > 0} ds \geq \sum_{k=1}^{D(t)} S_k.$$



Figure 9: A graphical ‘proof’ of  $Y = \lambda X$ . Here  $Y(t)/t \rightarrow Y$ ,  $n/T_n \rightarrow \lambda$  and  $n^{-1} \sum_{i=1}^n X_i \rightarrow X$ . Observe that in the figure  $X_k$  does not represent an inter-arrival time; instead it corresponds to the increment of (the graph of)  $Y(t)$  between two consecutive epochs  $T_{k-1}$  and  $T_k$  at which  $Y(t)$  is observed.

Explain this, and conclude that

$$\lambda E[S] \geq \rho \geq \delta E[S]. \quad (4.1.1)$$

Hence, if the system is rate-stable,  $\delta = \lambda$ , and then  $\rho = \lambda E[S]$ .

Hints

**h.4.1.1.** Define  $Y(t) = \int_0^t \mathbb{1}_{L(s)>0} ds$  as the total amount of time the server has been busy up to the time  $t$ . Then take as epochs  $T_k = D_k$  and use rate stability.

Solutions

**s.4.1.1.** It is evident that  $X_k = Y(D_k) - Y(D_{k-1}) = S_k$ , hence  $X = \lim_{n \rightarrow \infty} n^{-1} \sum_{k=1}^n X_k = E[S]$ . Also  $\lim_{t \rightarrow \infty} Y(t)/t = \rho$ . Finally, in the relation  $Y = \lambda X$ , the  $\lambda$  is  $\delta$  since we consider departure epochs  $T_k = D_k$ , rather than  $A_k$ . By the renewal reward theorem  $Y = \lambda X$  we get that  $\rho = \delta E[S]$ . Finally, by rate-stability, the job arrival rate  $\lambda = \delta$ , hence  $\rho = \lambda E[S]$ .

**s.4.1.2.** Observe that since  $t$  can lie half way a service interval and  $A(t) \geq D(t)$ . As  $A(t) \rightarrow \infty$  as  $t \rightarrow \infty$ ,

$$\lim_{t \rightarrow \infty} \frac{1}{t} \sum_{k=1}^{A(t)} S_k = \lim_{t \rightarrow \infty} \frac{A(t)}{t} \frac{1}{A(t)} \sum_{k=1}^{A(t)} S_k = \lim_{t \rightarrow \infty} \frac{A(t)}{t} \cdot \lim_{t \rightarrow \infty} \frac{1}{A(t)} \sum_{k=1}^{A(t)} S_k = \lambda E[S].$$

Applying similar limits to the other inequality gives the

Note that this is in fact the same argument as that underlies the renewal reward theorem. Henceforth we will just use the renewal reward theorem.

## 4.2 LEVEL CROSSING AND BALANCE EQUATIONS

Let us say that the system is in *state*  $n$  at time  $t$  when it contains  $n$  jobs at that moment, i.e., when  $L(t) = n$ . The system *up-crosses level*  $n$  at time  $t$  when its state changes from  $n$  to  $n + 1$

due to an arrival, and it *down-crosses* level  $n$  when its state changes from  $n + 1$  to  $n$  due to a departure. Clearly, the number of up-crossings and down-crossings cannot differ by more than 1, because it is only possible to up-cross level  $n$  after a down-crossing (or the other way around). This simple idea will prove a key stepping stone in the analysis of queueing systems.

To establish the section's main result (4.2.6), we need a few definitions that are quite subtle and might seem a bit abstract, but below we will provide intuitive interpretations in terms of system KPIs. After this, we will generalize the principle of level-crossing to *balance equations* which allow us to deal with more general types of transitions. We refer to Fig. 13 in which all concepts developed here are summarized.

### Theory and Exercises

LEVEL CROSSING    Define

$$A(n, t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t} \mathbb{1}_{L(A_k-) = n} \quad (4.2.1a)$$

as the number of arrivals up to time  $t$  that saw  $n$  customers in the system upon their arrival, cf. Fig. 10.

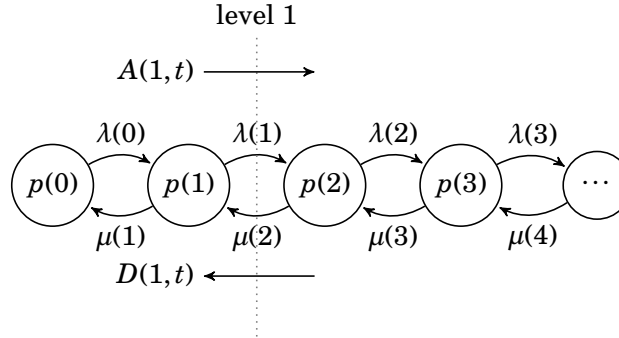


Figure 10:  $A(1, t)$  counts the number of jobs up to time  $t$  that saw 1 job in the system upon arrival, and right after such arrivals the system contains 2 jobs. Thus, each time  $A(1, t)$  increases by one, level 1 (the dotted line separating states 1 and 2) is crossed from below. Similarly,  $D(1, t)$  counts the number of departures that leave 1 job behind, and just before such departures the system contains 2 jobs. Hence, level 1 is crossed from above. It is evident that the number of times this level is crossed from below must be the same (plus or minus 1) as the number of times it is crossed from above. (We introduce  $\lambda(n)$ ,  $\mu(n)$  and  $p(n)$  below.)

**4.2.1.** Why do we take  $L(A_k-) = n$  rather than  $L(A_k)$  in the definition of  $A(n, t)$ ?

**4.2.2.** If  $A(n, t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t} \mathbb{1}_{L(A_k-) = n}$ , is  $A(t) = \sum_{n=0}^{\infty} A(n, t)$ ?

**4.2.3.** Show that  $A(n, t) \leq A(t)$ .

**4.2.4.** If  $\lambda > \delta$  can it happen that  $\lim_{t \rightarrow \infty} A(n, t)/t > 0$  for some (finite)  $n$ ?

Next, let

$$Y(n, t) = \int_0^t \mathbb{1}_{L(s) = n} ds \quad (4.2.1b)$$

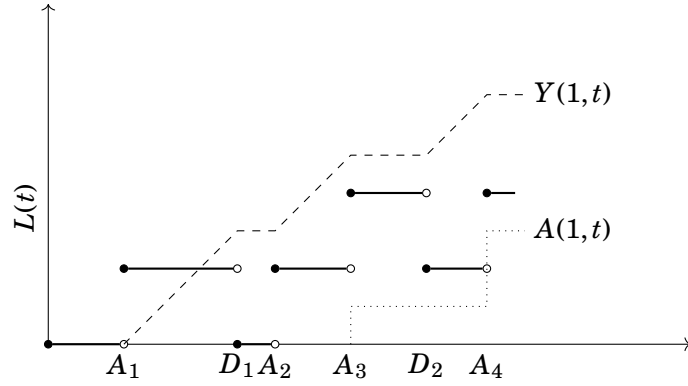


Figure 11: Plots of  $Y(1, t)$  and  $A(1, t)$ . (For visual clarity, we subtracted  $1/2$  from  $A(1, t)$ , for otherwise its graph would partly overlap with the graph of  $Y$ .)

be the total time the system contains  $n$  jobs during  $[0, t]$ , and

$$p(n, t) = \frac{1}{t} \int_0^t \mathbb{1}_{L(s)=n} ds = \frac{Y(n, t)}{t}, \quad (4.2.1c)$$

be the fraction of time that  $L(s) = n$  in  $[0, t]$ . Figure 11 illustrates the relation between  $Y(n, t)$  and  $A(n, t)$ .

**4.2.5.** Consider the following (silly) queueing process. At times  $0, 2, 4, \dots$  customers arrive, each customer requires 1 unit of service, and there is one server. Find an expression for  $A(n, t)$ . (What acronym would describe this queueing situation?)

**4.2.6** (Continuation of [4.2.5]). Find an expression for  $Y(n, t)$ .

Define also the limits:

$$\lambda(n) = \lim_{t \rightarrow \infty} \frac{A(n, t)}{Y(n, t)}, \quad p(n) = \lim_{t \rightarrow \infty} p(n, t), \quad (4.2.2)$$

as the *arrival rate in state  $n$*  and the *long-run fraction of time the system spends in state  $n$* . To clarify the former definition, observe that  $A(n, t)$  counts the number of arrivals that see  $n$  jobs in the system upon arrival, while  $Y(n, t)$  tracks the amount of time the system contains  $n$  jobs. Suppose that at time  $T$  a job arrives that sees  $n$  jobs in the system. Then  $A(n, T) = A(n, T-) + 1$ , and this job finishes an interval that is tracked by  $Y(n, t)$ , precisely because this job sees  $n$  jobs in the system just prior to its arrival. Thus, just as  $A(t)/t$  is the total number of arrivals during  $[0, t]$  divided by  $t$ ,  $A(n, t)/Y(n, t)$  is the number of arrivals that see  $n$  jobs divided by the time the system contains  $n$  jobs.

**4.2.7** (Continuation of [4.2.6]). Compute  $p(n)$  and  $\lambda(n)$ .

Similar to the definition for  $A(n, t)$ , let

$$D(n, t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \leq t} \mathbb{1}_{L(D_k)=n}$$

denote the number of departures up to time  $t$  that leave  $n$  customers behind. Then, define

$$\mu(n+1) = \lim_{t \rightarrow \infty} \frac{D(n, t)}{Y(n+1, t)},$$

as the departure rate from state  $n + 1$ . (It is easy to get confused here: to leave  $n$  jobs behind, the system must contain  $n + 1$  jobs just prior to the departure.) Figure 10 shows how  $A(n, t)$  and  $\lambda(n)$  relate to  $D(n, t)$  and  $\mu(n + 1)$ .

**4.2.8.** *Should we take  $D(n - 1, t)$  or  $D(n, t)$  in the definition of  $\mu(n)$ ?*

**4.2.9** (Continuation of [4.2.7]). *Compute  $D(n, t)$  and  $\mu(n + 1)$  for  $n \geq 0$ .*

Observe that customers arrive and depart as single units. Thus, if  $\{T_k\}$  is the ordered set of arrival and departure times of the customers, then  $L(T_k) = L(T_k -) \pm 1$ . But then we must also have that

$$|A(n, t) - D(n, t)| \leq 1, \quad (4.2.3)$$

(think about this). From this observation it follows immediately that

$$\lim_{t \rightarrow \infty} \frac{A(n, t)}{t} = \lim_{t \rightarrow \infty} \frac{D(n, t)}{t}. \quad (4.2.4)$$

With this equation we can obtain two nice and fundamental identities. The first we develop now; the second follows in Section 4.3.

The rate of jobs that ‘see the system with  $n$  jobs’ can be defined as  $A(n, t)/t$ . Taking limits we get

$$\lim_{t \rightarrow \infty} \frac{A(n, t)}{t} = \lim_{t \rightarrow \infty} \frac{A(n, t)}{Y(n, t)} \frac{Y(n, t)}{t} = \lambda(n)p(n), \quad (4.2.5a)$$

where we use the above definitions for  $\lambda(n)$  and  $p(n)$ . Similarly, the departure rate of jobs that leave  $n$  jobs behind is

$$\lim_{t \rightarrow \infty} \frac{D(n, t)}{t} = \lim_{t \rightarrow \infty} \frac{D(n, t)}{Y(n + 1, t)} \frac{Y(n + 1, t)}{t} = \mu(n + 1)p(n + 1). \quad (4.2.5b)$$

Combining this with (4.2.4) we arrive at the *level-crossing equations*

$$\lambda(n)p(n) = \mu(n + 1)p(n + 1). \quad (4.2.6)$$

**4.2.10** (Continuation of [4.2.9]). *Compute  $\lambda(n)p(n)$  for  $n \geq 0$ , and check  $\lambda(n)p(n) = \mu(n + 1)p(n + 1)$ .*

Result (4.2.6) turns out to be exceedingly useful, as will become evident from Section 5.1 onward. More specifically, by specifying (i.e., modeling)  $\lambda(n)$  and  $\mu(n)$ , we can compute the long-run fraction of time  $p(n)$  that the system contains  $n$  jobs. To see this, rewrite the above into

$$p(n + 1) = \frac{\lambda(n)}{\mu(n + 1)} p(n). \quad (4.2.7)$$

Thus, this equation fixes the ratios between the probabilities. In other words, if we know  $p(n)$  we can compute  $p(n + 1)$ , and so on. Hence, if  $p(0)$  is known, then  $p(1)$  follows, from which  $p(2)$  follows, and so on. A straightaway iteration then leads to

$$p(n + 1) = \frac{\lambda(n)\lambda(n - 1) \cdots \lambda(0)}{\mu(n + 1)\mu(n) \cdots \mu(1)} p(0). \quad (4.2.8)$$

To determine  $p(0)$  we can use the fact that the numbers  $p(n)$  represent probabilities. Hence, from the normalizing condition  $\sum_{n=0}^{\infty} p(n) = 1$ , we get  $p(0) = G^{-1}$  with  $G$  being the *normalization constant*

$$G = 1 + \sum_{n=0}^{\infty} \frac{\lambda(n)\lambda(n - 1) \cdots \lambda(0)}{\mu(n + 1)\mu(n) \cdots \mu(1)}. \quad (4.2.9)$$

In the next few sections we will make suitable choices for  $\lambda(n)$  and  $\mu(n)$  to model many different queueing situations so that, based on (4.2.6), we can obtain simple expressions for  $p(n)$  in terms of the arrival and service rates.

With  $p(n)$  we define two easy, but important performance measures. The time-average number of items in the system becomes

$$E[L] = \sum_{n=0}^{\infty} np(n),$$

and the long-run fraction of time the system contains at least  $n$  jobs is

$$P(L \geq n) = \sum_{i=n}^{\infty} p(i).$$

**4.2.11.** Derive  $E[L] = \sum_{n=0}^{\infty} np(n)$  from (2.4.6).

Finally, the following two exercises show that level-crossing arguments extend well beyond the queueing systems modeled by Fig. 10.

**4.2.12.** Consider a single server that serves one queue and serves only in batches of 2 jobs at a time (so never 1 job or more than 2 jobs), i.e., the  $M/M^2/1/3$  queue. Single jobs arrive at rate  $\lambda$  and the inter-arrival times are exponentially distributed so that we can assume that  $\lambda(n) = \lambda$ . The batch service times are exponentially distributed with mean  $1/\mu$ . Then, by the memoryless property,  $\mu(n) = \mu$ . At most 3 jobs fit in the system. Make a graph of the state-space and show, with arrows, the transitions that can occur.

**4.2.13.** Use the graph of [4.2.12] and a level-crossing argument to express the steady-state probabilities  $p(n), n = 0, \dots, 3$  in terms of  $\lambda$  and  $\mu$ .

**INTERPRETATION** The definitions in (4.2.1) may seem a bit abstract, but they obtain an immediate interpretation when relating them to applications. To see this, we discuss two examples.

Consider the sorting process of post parcels at a distribution center of a post-delivery company. Each day tens of thousands of incoming parcels have to be sorted to their final destination. In the first stage of the process, parcels are sorted to a region in the Netherlands. Incoming parcels are deposited on a conveyor belt. From the belt, they are carried to outlets (chutes), each chute corresponding to a specific region. Employees take out the parcels from the chutes and put the parcels in containers. The arrival rate of parcels for a certain chute may temporarily exceed the working capacity of the employees, as such the chute serves as a queue. When the chute overflows, parcels are directed to an overflow container and are sorted the next day. The target of the sorting center is to deliver at least a certain percentage of the parcels within one day. Thus, the fraction of parcels rejected at the chute should remain small.

Suppose a chute can contain at most 20 parcels, say. Then, each parcel on the belt that ‘sees’ 20 parcels in its chute will be blocked. Let  $L(t)$  be the number of parcels in the chute at time  $t$ . Then,  $A(20, t)$  as defined in (4.2.1a) is the number of *blocked parcels* up to time  $t$ , and  $A(20, t)/A(t)$  is the fraction of rejected parcels. In fact,  $A(20, t)$  and  $A(t)$  are continuously tracked by the sorting center and used to adapt employee capacity to control the fraction of rejected parcels. Thus, in simulations, if one wants to estimate loss fractions,  $A(n, t)/A(t)$  is the most natural concept to consider.



For the second example, suppose there is a cost associated with keeping jobs in queue. Let  $w$  be the cost per job in queue per unit time so that the cost rate is  $nw$  when  $n$  jobs are in queue. But then  $wnY(n, t)$  is the total cost up to time  $t$  to have  $n$  jobs in queue, hence the total cost up to time  $t$  is

$$C(t) = w \sum_{n=0}^{\infty} nY(n, t),$$

and the average cost is

$$\frac{C(t)}{t} = w \sum_{n=0}^{\infty} n \frac{Y(n, t)}{t} = w \sum_{n=0}^{\infty} np(n, t).$$

All in all, the concepts developed above have natural interpretations in practical queueing situations; they are useful in theory and in simulation, as they relate the theoretical concepts to actual measurements.

**BALANCE EQUATIONS** It is important to realize that the level-crossing argument cannot always be used as we do here. The reason is that sometimes there does not exist a line between two states such that the state space splits into two disjoint parts. For a more general approach, we focus on a single state and count how often this state is entered and left, cf. Fig. 12. Specifically, define

$$I(n, t) = A(n-1, t) + D(n, t),$$

as the number of times the queueing process enters state  $n$  either due to an arrival from state  $n-1$  or due to a departure leaving  $n$  jobs behind. Similarly,

$$O(n, t) = A(n, t) + D(n-1, t),$$

counts how often state  $n$  is left either by an arrival (to state  $n+1$ ) or a departure (to state  $n-1$ ).

Of course,  $|I(n, t) - O(n, t)| \leq 1$ . Thus, from the fact that

$$\lim_{t \rightarrow \infty} \frac{I(n, t)}{t} = \lim_{t \rightarrow \infty} \frac{A(n-1, t)}{t} + \lim_{t \rightarrow \infty} \frac{D(n, t)}{t} = \lambda(n-1)p(n-1) + \mu(n+1)p(n+1)$$

and

$$\lim_{t \rightarrow \infty} \frac{O(n, t)}{t} = \lim_{t \rightarrow \infty} \frac{A(n, t)}{t} + \lim_{t \rightarrow \infty} \frac{D(n-1, t)}{t} = \lambda(n)p(n) + \mu(n)p(n)$$

we get that

$$\lambda(n-1)p(n-1) + \mu(n+1)p(n+1) = (\lambda(n) + \mu(n))p(n).$$

These equations hold for any  $n \geq 0$  and are known as the *balance equations*. We will use these equations when studying queueing systems in which level-crossing cannot be used, for instance for queueing networks.

Again, just by using properties, i.e., counting differences, that hold along any sensible sample path we obtain very useful statistical and probabilistic results.

### Hints

**h.4.2.1.** Recall that  $L(t)$  is *right-continuous*.

**h.4.2.5.** For the acronym, observe that the service times and inter-arrival are deterministic and there is one server. For the computation of  $Y(n, t)$ , make a plot of  $L(s)$  as a function of time for  $n = 1$ .

Make a plot of  $L(s)$  for  $n = 1$  as a function of time.

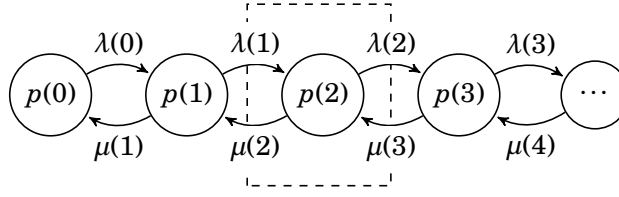


Figure 12: For the balance equations we count how often a box around a state is crossed from inside and outside. In the long run, the entering and leaving rates should be equal. For the example here, the rate out is  $p(2)\lambda(2) + p(2)\mu(2)$  while the rate in is  $p(1)\lambda(1) + p(3)\mu(3)$ .

**h.4.2.13.** First, balance the rates across the levels. Then solve this in terms of  $p(0)$ .

*Solutions*

**s.4.2.1.**  $L(t)$  is the number of customers in the system at time  $t$ . As such the function  $t \rightarrow L(t)$  is *right-continuous*. The definition of  $L(A_k-) = \lim_{t \uparrow A_k} L(t)$  is the limit from the left. The customer therefore ‘sees’  $L(A_k-)$  just before he/she arrives.

**s.4.2.2.**  $A(t)$  counts all customers that arrive up to time  $t$ , i.e., during  $[0, t]$ . Note that this *includes* time  $t$ .  $A(n, t)$  counts the jobs that see  $n$  jobs in the system just before they arrive. As there are either 0, or 1, or 2, etc., jobs in the system upon arrival the summation covers all possible outcomes. Therefore the claim is true.

**s.4.2.3.** Observe that  $\mathbb{1}_{A_k \leq t} \mathbb{1}_{L(A_k-) = n} \leq \mathbb{1}_{A_k \leq t}$ ; the last inequality follows from the fact that  $\mathbb{1}_{L(A_k-) = n} \leq 1$ . Therefore,

$$A(n, t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t} \mathbb{1}_{L(A_k-) = n} \leq \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t} = A(t).$$

For any ‘normal’ queueing system,  $A(t) > A(n, t)$ , because the queue length fluctuates.

**s.4.2.4.** If  $\lambda > \delta$ , then  $L(t) \rightarrow \infty$ . But then there must be a last time,  $s$  say, that  $L(s) = n + 1$ , and  $L(t) > n + 1$  for all  $t > s$ . Hence, after time  $s$  no job will see the system with  $n$  jobs. Thus  $A(n, t) = A(n, s)$  for all  $t > s$ . This is a finite number, while  $t \rightarrow \infty$ , so that  $A(n, t)/t \rightarrow 0$ .

**s.4.2.5.** It is the  $D/D/1$  queue, since there is one server and the inter-arrival times and service times are constant, i.e., deterministic.  $A_k = 2(k - 1)$  as jobs arrive at  $t = 0, 2, 4, \dots$ , hence,  $A(t) \approx t/2$  when  $t \gg 0$ . We also know that  $L(s) = 1$  if  $s \in [2i, 2i + 1)$  and  $L(s) = 0$  for  $s \in [2i - 1, 2i)$  for  $i = 0, 1, 2, \dots$ . Thus,  $L(A_k-) = L(2k-) = 0$ . Hence,  $A(0, t) \approx t/2$  for  $t \gg 0$ , and  $A(n, t) = 0$  for  $n \geq 1$ .

**s.4.2.6.** Next, to get  $Y(n, t)$ , observe that the system never contains more than 1 job. Hence,  $Y(n, t) = 0$  for all  $n \geq 2$ . Then we see that  $Y(1, t) = \int_0^t \mathbb{1}_{L(s)=1} ds$ . Now observe that for our queueing system  $L(s) = 1$  for  $s \in [0, 1)$ ,  $L(s) = 0$  for  $s \in [1, 2)$ ,  $L(s) = 1$  for  $s \in [2, 3)$ , and so on. Thus, when  $t < 1$ ,  $Y(1, t) = \int_0^t \mathbb{1}_{L(s)=1} ds = \int_0^t 1 ds = t$ . When  $t \in [1, 2)$ ,

$$L(t) = 0 \implies \mathbb{1}_{L(t)=0} \implies Y(1, t) \text{ does not change.}$$

Continuing to  $[2, 3)$  and so on gives

$$Y(1, t) = \begin{cases} t & t \in [0, 1), \\ 1 & t \in [1, 2), \\ 1 + (t - 2) & t \in [2, 3), \\ 2 & t \in [3, 4), \\ 2 + (t - 4) & t \in [4, 5), \end{cases}$$

and so on. Since  $Y(n, t) = 0$  for all  $n \geq 2$ ,  $L(s) = 1$  or  $L(s) = 0$  for all  $s$ , therefore,

$$Y(0, t) = t - Y(1, t).$$

**s.4.2.7.** From the other exercises:

$$\begin{aligned} \lambda(0) &\approx \frac{A(0, t)}{Y(0, t)} \approx \frac{t/2}{t/2} = 1, \\ \lambda(1) &\approx \frac{A(1, t)}{Y(1, t)} \approx \frac{0}{t/2} = 0, \\ p(0) &\approx \frac{Y(0, t)}{t} \approx \frac{t/2}{t} = \frac{1}{2}, \\ p(1) &\approx \frac{Y(1, t)}{t} \approx \frac{t/2}{t} = \frac{1}{2}. \end{aligned}$$

For the rest  $\lambda(n) = 0$ , and  $p(n) = 0$ , for  $n \geq 2$ .

**s.4.2.8.**  $D(n - 1, t)$  counts the departures that leave  $n - 1$  behind. Thus, just before the customer leaves, the system contains  $n$  customers.

**s.4.2.9.**  $D(0, t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \leq t, L(D_k)=0}$ . From the graph of  $\{L(s)\}$  we see that all jobs leave an empty system behind. Thus,  $D(0, t) \approx t/2$ , and  $D(n, t) = 0$  for  $n \geq 1$ . With this,  $D(0, t)/Y(1, t) \sim (t/2)/(t/2) = 1$ , and so,

$$\mu(1) = \lim_{t \rightarrow \infty} \frac{D(0, t)}{Y(1, t)} = 1,$$

and  $\mu(n) = 0$  for  $n \geq 2$ .

**s.4.2.10.**  $\lambda(0)p(0) = 1 \cdot 1/2 = 1/2$ ,  $\lambda(n)p(n) = 0$  for  $n > 1$ , as  $\lambda(n) = 0$  for  $n > 0$ .

From [4.2.9],  $\mu(1) = 1$ , hence  $\mu(1)p(1) = 1 \cdot 1/2 = 1/2$ . Moreover,  $\mu(n) = 0$  for  $n \geq 2$ .

Clearly, for all  $n$  we have  $\lambda(n)p(n) = \mu(n + 1)p(n + 1)$ .

**s.4.2.11.** As  $L(s)$  counts the number of jobs in the system at time  $s$  (thus  $L(s)$  is an integer),

$$L(s) = \sum_{n=0}^{\infty} n \mathbb{1}_{L(s)=n}.$$

With this we can write for the time-average number of jobs in the system

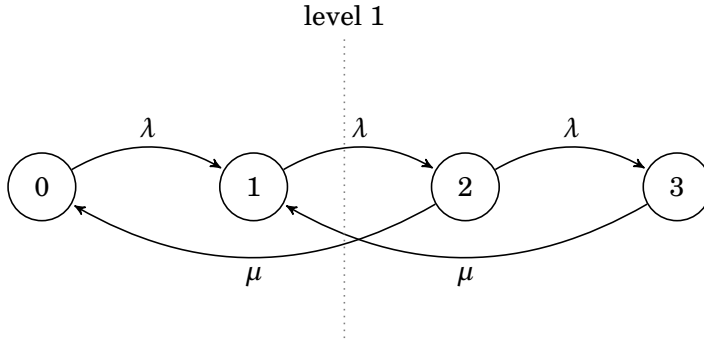
$$\frac{1}{t} \int_0^t L(s) ds = \frac{1}{t} \int_0^t \left( \sum_{n=0}^{\infty} n \mathbb{1}_{L(s)=n} \right) ds = \sum_{n=0}^{\infty} \frac{n}{t} \int_0^t \mathbb{1}_{L(s)=n} ds, \quad (4.2.10)$$

where we interchange the integral and the summation<sup>1</sup>. It then follows from (4.2.1c) that

$$\frac{1}{t} \int_0^t L(s) ds = \sum_{n=0}^{\infty} n p(n, t).$$

Finally, assuming that the limit  $p(n, t) \rightarrow p(n)$  exists as  $t \rightarrow \infty$  (and that the summation and limit can be interchanged in the above), the result follows.

**s.4.2.12.** See the figure below.



**s.4.2.13.** With level-crossing:

$$\begin{aligned} \lambda p(0) &= \mu p(2), & \text{the level between 0 and 1,} \\ \lambda p(1) &= \mu p(2) + \mu p(3), & \text{see level 1,} \\ \lambda p(2) &= \mu p(3), & \text{the level between 2 and 3.} \end{aligned}$$

Solving this in terms of  $p(0)$  gives  $p(2) = \rho p(0)$ ,  $p(3) = \rho p(2) = \rho^2 p(0)$ , and

$$\lambda p(1) = \mu(p(2) + p(3)) = \mu(\rho + \rho^2)p(0) = (\lambda + \lambda^2/\mu)p(0),$$

hence  $p(1) = p(0)(\mu + \lambda)/\mu$ .

### 4.3 POISSON ARRIVALS SEE TIME AVERAGES

#### *Theory and Exercises*

Suppose the following limit exists:

$$\pi(n) = \lim_{m \rightarrow \infty} \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{L(A_k-) = n}, \quad (4.3.1)$$

then  $\pi(n)$  is the long-run fraction of jobs that, upon arrival, observe  $n$  customers in the system. It is natural to ask whether  $\pi(n)$  and  $p(n)$ , as defined by (4.2.2), are related, that is, whether what customers see upon arrival is related to the time-average behavior of the system. In this section we will derive the famous *Poisson arrivals see time averages (PASTA)* condition that ensures that  $\pi(n) = p(n)$  if jobs arrive in accordance with a Poisson process.

<sup>1</sup> This is allowed as the integrand is non-negative. More generally, the interested reader should check Fubini's theorem.

Since  $A(t) \rightarrow \infty$  as  $t \rightarrow \infty$ , it is reasonable that (see [4.3.7] for a proof)

$$\begin{aligned}\pi(n) &= \lim_{t \rightarrow \infty} \frac{1}{A(t)} \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-) = n} = \lim_{t \rightarrow \infty} \frac{1}{A(t)} \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t, L(A_k-) = n} \\ &= \lim_{t \rightarrow \infty} \frac{A(n, t)}{A(t)},\end{aligned}\tag{4.3.2}$$

where we use (4.2.1a) in the last row. But, with (2.3.1),

$$\frac{A(n, t)}{t} = \frac{A(t)}{t} \frac{A(n, t)}{A(t)} \rightarrow \lambda \pi(n), \quad \text{as } t \rightarrow \infty,\tag{4.3.3}$$

while by (4.2.5),

$$\frac{A(n, t)}{t} = \frac{A(n, t)}{Y(n, t)} \frac{Y(n, t)}{t} \rightarrow \lambda(n) p(n), \quad \text{as } t \rightarrow \infty.$$

Thus

$$\lambda \pi(n) = \lambda(n) p(n).\tag{4.3.4}$$

This leads to our final result:

$$\lambda(n) = \lambda \iff \pi(n) = p(n).$$

This means that if the arrival rate does not depend on the state of the system, i.e.,  $\lambda(n) = \lambda$ , the sample probabilities  $\{\pi(n)\}$  are equal to the time-average probabilities  $\{p(n)\}$ . In other words, the customer perception at arrival moments is the same as the server perception.

As the next exercises show, this property is not satisfied in general. However, when the arrival process is Poisson we have that  $\lambda(n) = \lambda$ . This fact is called *PASTA: Poisson Arrivals See Time Averages*.

**4.3.1.** Show for the case of [4.2.5] that  $\pi(0) = 1$  and  $\pi(n) = 0$ , for  $n > 0$ .

**4.3.2.** Check that (4.3.4) holds for the system of [4.3.1].

With the above reasoning, we can also establish a relation between  $\pi(n)$  and the statistics of the system as obtained by the departures. Define, analogous to (4.3.2),

$$\delta(n) = \lim_{t \rightarrow \infty} \frac{D(n, t)}{D(t)}\tag{4.3.5}$$

as the long-run fraction of jobs that leave  $n$  jobs behind. From (4.2.4),

$$\frac{A(t)}{t} \frac{A(n, t)}{A(t)} = \frac{A(n, t)}{t} \approx \frac{D(n, t)}{t} = \frac{D(t)}{t} \frac{D(n, t)}{D(t)}.$$

Taking limits at the left and right, and using (2.3.2), we obtain for (queueing) systems in which customers arrive and leave as single units that

$$\lambda \pi(n) = \delta \delta(n).\tag{4.3.6}$$

Thus, if the system is rate-stable and transitions occur one-by-one, the statistics obtained by arrivals is the same as statistics obtained by departures, i.e.,

$$\lambda = \delta \iff \pi(n) = \delta(n).\tag{4.3.7}$$

**4.3.3.** For the G/G/1 queue, prove that the fraction of jobs that see  $n$  jobs in the system upon arrival is the same as the fraction of departures that leave  $n$  jobs behind. What condition have you used to prove this?

**4.3.4.** When  $\lambda \neq \delta$ , is  $\pi(n) \geq \delta(n)$ ?

**4.3.5.** Show that

$$\lambda\pi(n) = \lambda(n)p(n) = \mu(n+1)p(n+1) = \delta\delta(n).$$

What is the important condition for this to be true?

**4.3.6.** Use PASTA and the balance equations of the  $M/M/1$  queue to derive that  $(\lambda + \mu)\pi(n) = \lambda\pi(n-1) + \mu\pi(n+1)$ .

**4.3.7.** There is a subtle problem in the transition from (4.3.1) to (4.3.2) and the derivation of (4.3.3):  $\pi(n)$  is defined as a limit over arrival epochs while in  $A(n, t)/t$  we take the limit over time. Now the observant reader might ask why these limits should relate at all. Use the renewal reward theorem to show that (4.3.2) is valid.

*Hints*

**h.4.3.3.** Note that (4.2.3) does not depend on the distribution of the inter-arrival or service times.

**h.4.3.4.** Use that  $\lambda \geq \delta$  always holds. Thus, when  $\lambda \neq \delta$ , it must be that  $\lambda > \delta$ . What are the consequences of this inequality; how does the queue length behave as a function of time?

**h.4.3.5.** Check all definitions of  $Y(n, t)/t$  and so on.

**h.4.3.6.** Consider some state  $n$  (not a level) and count all transitions that ‘go in and out of’ this state. Specifically,  $A(n, t) + D(n-1, t)$  counts all transitions out of state  $n$ :  $A(n, t)$  counts the number of arrivals that see  $n$  in the system upon arrival, hence immediately after such arrivals the system contains  $n+1$  jobs; likewise,  $D(n-1, t)$  counts all jobs that leave  $n-1$  jobs behind, hence immediately before such jobs depart the system contains  $n$  jobs. In a similar way,  $A(n-1, t) + D(n, t)$  counts all transitions into state  $n$  (Recall once again,  $D(n, t)$  counts the jobs that leave  $n$  behind. Hence, when such departures occur, state  $n$  is entered). Now use that ‘what goes in must go out’.

**h.4.3.7.** Check that the conditions of the renewal reward theorem are satisfied in the above proof of (4.3.3). Then define

$$Y(t) := A(n, t) = \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-) = n}$$

$$X_k := Y(A_k) - Y(A_{k-1}) = A(n, A_k) - A(n, A_{k-1}) = \mathbb{1}_{L(A_k-) = n}.$$

*Solutions*

**s.4.3.1.** All arrivals see an empty system. Hence  $A(0, t)/A(t) \approx (t/2)/(t/2) = 1$ , and  $A(n, t) = 0$  for  $n > 0$ . Thus,  $\pi(0) = \lim_{t \rightarrow \infty} A(0, t)/A(t) = 1$  and  $\pi(n) = 0$  for  $n > 0$ . Recall from the other exercises that  $p(0) = 1/2$ . Hence, statistics as obtained via time averages are not necessarily the same as statistics obtained at arrival moments (or any other point process).

**s.4.3.2.** From the relevant previous exercises,  $\lambda = \lim_{t \rightarrow \infty} A(t)/t = 1/2$ .  $\lambda(0) = 1$ ,  $p(0) = 1/2$ , and  $\pi(0) = 1$ . Hence,

$$\lambda\pi(0) = \lambda(0)p(0) \implies \frac{1}{2} \times 1 = 1 \times \frac{1}{2}.$$

For  $n > 0$  it's easy, everything is 0.

**s.4.3.3.** All follows straightaway from the definitions in the main text. In the  $G/G/1$  queue jobs arrive and depart in single units. Then, from (4.2.3),

$$\frac{A(t)}{t} \frac{A(n,t)}{A(t)} \approx \frac{D(t)}{t} \frac{D(n,t)}{D(t)}.$$

The left-hand side goes to  $\lambda\pi(n)$  as  $t \rightarrow \infty$ , and the right-hand side to  $\delta\delta(n)$ . Use the fact that we always assume, implicitly, that the system is stable, so that  $\lambda = \delta$ . As a consequence  $\delta(n) = \pi(n)$  for the  $G/G/1$  queue.

**s.4.3.4.** The assumptions lead us to conclude that  $\lambda > \delta$ . As a consequence, the queue length must increase in the long run (jobs come in faster than they leave). Therefore,  $A(n,t)/t \rightarrow 0$  for all  $n$ , and also  $D(n,t)/t \rightarrow 0$ . Consequently,  $\pi(n) = \delta(n) = 0$ , which is the only sensible reconciliation with (4.3.6).

**s.4.3.5.** The important condition is that transitions occur as single steps. In other words, the relation is true for processes with *one-step transitions*, i.e., when  $|A(n,t) - D(n,t)| \leq 1$ . In that case,

$$\begin{aligned} \frac{A(n,t)}{t} &= \frac{A(n,t)}{A(t)} \frac{A(t)}{t} \rightarrow \pi(n)\lambda \\ \frac{A(n,t)}{t} &= \frac{A(n,t)}{Y(n,t)} \frac{Y(n,t)}{t} \rightarrow \lambda(n)p(n) \\ \frac{D(n,t)}{t} &= \frac{D(n,t)}{Y(n+1,t)} \frac{Y(n+1,t)}{t} \rightarrow \mu(n+1)p(n+1) \\ \frac{D(n,t)}{t} &= \frac{D(n,t)}{D(t)} \frac{D(t)}{t} \rightarrow \delta(n)\delta. \end{aligned}$$

**s.4.3.6.** By the hint, the difference between the ‘out transitions’ and the ‘in transitions’ is at most 1 for all  $t$ . Thus, we can write

$$\begin{aligned} \text{transitions out} &\approx \text{transitions in} \iff \\ A(n,t) + D(n-1,t) &\approx A(n-1,t) + D(n,t) \iff \\ \frac{A(n,t) + D(n-1,t)}{t} &\approx \frac{A(n-1,t) + D(n,t)}{t} \iff \\ \frac{A(n,t)}{t} + \frac{D(n-1,t)}{t} &\approx \frac{A(n-1,t)}{t} + \frac{D(n,t)}{t}. \end{aligned}$$

Using the ideas of Section 4.2 this becomes for  $t \rightarrow \infty$ ,

$$(\lambda(n) + \mu(n))p(n) = \lambda(n-1)p(n-1) + \mu(n+1)p(n+1).$$

Since we are concerned here with the  $M/M/1$  queue we have that  $\lambda(n) = \lambda$  and  $\mu(n) = \mu$ , and using PASTA we have that  $p(n) = \pi(n)$ . We are done.

**s.4.3.7.** First we check the conditions. The counting process here is  $\{A(t)\}$  and the epochs at which  $A(t)$  increases are  $\{A_k\}$ . By assumption,  $A_k \rightarrow \infty$ , hence  $A(t) \rightarrow \infty$  as  $t \rightarrow \infty$ . Moreover, by assumption  $A(t)/t \rightarrow \lambda$ . Also  $A(n, t)$  is evidently non-decreasing and  $A(n, t) \rightarrow \infty$  as  $t \rightarrow \infty$ .

From the definitions in the hint,

$$X = \lim_{m \rightarrow \infty} \frac{1}{m} \sum_{k=1}^m X_k = \lim_{m \rightarrow \infty} \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{L(A_k -) = n} = \pi(n).$$

Since  $Y = \lim_{t \rightarrow \infty} Y(t)/t = \lim_{t \rightarrow \infty} A(n, t)/t$  it follows from the renewal reward theorem that

$$Y = \lambda X \implies \lim_{t \rightarrow \infty} \frac{A(n, t)}{t} = \lambda X = \lambda \pi(n).$$

Thus, (4.3.3) follows from the renewal reward theorem.

#### 4.4 LITTLE'S LAW

There is an important relation between the average time  $E[W]$  a job spends in the system and the long-run time-average number  $E[L]$  of jobs that is contained in the system, which is called *Little's law*:

$$E[L] = \lambda E[W]. \quad (4.4.1)$$

Part of the usefulness of Little's law is that it applies under very general conditions to all input-output systems, whether the system is a queueing system or an inventory system or some much more general system. Hence, we will apply Little's law often in the forthcoming sections. The aim of this section is to prove this law.

##### *Theory and Exercises*

We start by defining a few intuitively useful concepts. From (1.4.10), we see that

$$\frac{1}{t} \int_0^t L(s) ds = \frac{1}{t} \int_0^t (A(s) - D(s)) ds$$

is the time-average of the number of jobs in the system during  $[0, t]$ . Next, the waiting time of the  $k$ th job is the time between the moment the job arrives and departs, that is,

$$W_k = \int_0^\infty \mathbb{1}_{A_k \leq s < D_k} ds.$$

Fig. 4 relates  $W_k$  to  $L(t)$ .

Consider a departure time  $T$  at which the system is empty so that  $A(T) = D(T)$ . Then, for  $k \leq A(T)$ ,

$$W_k = \int_0^T \mathbb{1}_{A_k \leq s < D_k} ds,$$

and for  $s \leq T$ ,

$$L(s) = \sum_{k=1}^\infty \mathbb{1}_{A_k \leq s < D_k} = \sum_{k=1}^{A(T)} \mathbb{1}_{A_k \leq s < D_k}.$$

**4.4.1.** Show that

$$\int_0^T L(s) ds = \sum_{k=1}^{A(T)} W_k.$$



**4.4.2.** Observe that the area between the graphs of  $A(s)$  and  $D(s)$  must be equal to the total waiting time spent by all jobs in the system until  $T$ . Use this to provide a graphical interpretation of the proof of Little's law.

**4.4.3.** Prove Little's law under the assumptions that  $A(T_i) = D(T_i)$  for an infinite number of times  $\{T_i\}$  such  $T_i \rightarrow \infty$  and that all limits exist.

**4.4.4.** Use the (physical) dimensions of the components of Little's law to check that  $E[W] \neq \lambda E[L]$ . (With this check, you can prevent making an often-made mistake.)

**4.4.5.** Consider the server of the  $G/G/1$  queue as a system by itself. The time jobs stay in this system is  $E[S]$ , and jobs arrive at rate  $\lambda$ . Use Little's law to conclude that  $\lambda E[S] = \rho := \lim_{t \rightarrow \infty} t^{-1} \int_0^t L_S(s) ds$ .

**4.4.6.** For a given single-server queueing system the average number of customers in the system is  $E[L] = 10$ , customers arrive at rate  $\lambda = 5$  per hour and are served at rate  $\mu = 6$  per hour. What is the average time customers spend in the system?

**4.4.7.** For a given single-server queueing system the average number of customers in the system is  $E[L] = 10$ , customers arrive at rate  $\lambda = 5$  per hour and are served at rate  $\mu = 6$  per hour. Suppose that at the moment you join the system, the number of customers in the system is 10. What is your expected time in the system?

With the PASTA property and Little's law it becomes quite easy to derive expressions for the average queue length and waiting times for the  $M/M/1$  queue. The average waiting time  $E[W]$  in the entire system is the expected time in queue plus the expected time in service, i.e.,

$$E[W] = E[W_Q] + E[S]. \quad (4.4.2)$$

By the PASTA property we have for the  $M/M/1$  queue that

$$E[W_Q] = E[L] E[S]. \quad (4.4.3)$$

**4.4.8.** Use Little's law to show for the  $M/M/1$  queue that

$$\begin{aligned} E[W] &= \frac{E[S]}{1-\rho}, & E[L] &= \frac{\rho}{1-\rho}, \\ E[L_Q] &= \frac{\rho^2}{1-\rho}, & E[L_s] &= \rho. \end{aligned}$$

**4.4.9.** Why is (4.4.3) not true in general for the  $M/G/1$  queue?

In the above proof of Little's law we assumed that there is a sequence of moments  $\{T_k, k = 0, 1, \dots\}$  at which the system is empty and such that  $T_k < T_{k+1}$  and  $T_k \rightarrow \infty$ . However, in many practical queueing situations the system is never empty. Thus, to be able to apply Little's law to such more general situations we should slacken the assumption that such a sequence exists. The aim of this set of questions is to find an educated guess for a more general assumption under which Little's law can hold.

**4.4.10.** Motivate in words (or with a derivation, if you prefer this) why the following is true:

$$\sum_{k=1}^{A(t)} W_k \geq \int_0^t L(s) ds \geq \sum_{k=1}^{D(t)} W_k.$$

**4.4.11.** Take suitable limits (and assume all these limits exist) to show that

$$\lambda E[W] \geq E[L] \geq \delta E[W].$$

Make explicit all the points where you use the strong law of large numbers.

**4.4.12.** Suppose that  $A(t) = \lambda t$  and  $D(t) = [A(t) - 10]^+$ . Explain that for this system the above assumption on  $\{T_k\}$  is violated. Show that Little's law is still true.

**4.4.13.** Based on the above formulate an educated guess for more general conditions under which Little's law holds. (You don't have to prove Little's law under your condition; postpone that to after the exam.)

*Hints*

**h.4.4.1.** Substitute the definition of  $L(s)$  in the left-hand side, then reverse the integral and summation.

**h.4.4.2.** Make a drawing of  $A(t)$  and  $D(t)$  until time  $T$ , i.e., the first time the system is empty. Observe that  $A(t) - D(t)$  is the number of jobs in the system. Take some level  $k$ , and compute  $A_k = A^{-1}(k)$  and  $D_k = D^{-1}(k)$ . Observe that  $D_k - A_k = D^{-1}(k) - A^{-1}(k)$  is the waiting time of job  $k$ .

**h.4.4.3.** In the result of [4.4.1], divide both sides by  $T$ . At the right-hand side use that  $1/T = A(T)/T \cdot 1/A(T)$ . Take limits.

**h.4.4.4.** Checking the dimensions in the formula prevents painful mistakes.

**h.4.4.6.** Start with checking the units when applying Little's law.

**h.4.4.8.** Combine (4.4.2) and (4.4.3) and apply Little's law.

*Solutions*

**s.4.4.1.**

$$\begin{aligned} \int_0^T L(s) \, ds &= \int_0^T \sum_{k=1}^{A(T)} \mathbb{1}_{A_k \leq s < D_k} \, ds \\ &= \sum_{k=1}^{A(T)} \int_0^T \mathbb{1}_{A_k \leq s < D_k} \, ds = \sum_{k=1}^{A(T)} W_k. \end{aligned}$$

**s.4.4.2.** The area enclosed between the graphs of  $A(t)$  and  $D(t)$  until  $T$  can be 'chopped up' in two ways: in the horizontal and in the vertical direction. (Please make the drawing as you go along. . . ) A horizontal line between  $A(t)$  and  $D(t)$  corresponds to the waiting time of a job, while a vertical line corresponds to the number of jobs in the system at time  $t$ . Now adding all horizontal lines (by integrating along the  $y$ -axis) makes up the total amount of waiting done by all the jobs until time  $T$ . On the other hand, adding the vertical lines (by integrating along the

$x$ -axis) is equal to the summation of all jobs in the system. Since the area is the same no matter whether you sum it in the horizontal or vertical direction:

$$\sum_{k=1}^{A(T)} W_k = \text{enclosed area} = \int_0^T (A(t) - D(t)) dt.$$

Dividing both sides by  $A(T)$  gives

$$\frac{1}{A(T)} \sum_{k=1}^{A(T)} W_k = \frac{1}{A(T)} \int_0^T (A(t) - D(t)) dt.$$

Finally, observe that this equality holds between any two times  $T_i, T_{i+1}$ , where times  $\{T_i\}$  are such that  $A(T_i) = D(T_i)$ . Then, as  $T_i \rightarrow \infty$ , which we assumed from the on-set,  $\frac{1}{A(T_i)} \sum_{k=1}^{A(T_i)} W_k \rightarrow E[W]$ , and

$$\frac{T_i}{A(T_i)} \frac{1}{T_i} \int_0^{T_i} (A(t) - D(t)) dt \rightarrow \lambda^{-1} E[L].$$

Hence, Little's law follows.

**s.4.4.3.** First solve [4.4.1]. Then,

$$\frac{1}{T} \int_0^T L(s) ds = \frac{A(T)}{T} \frac{1}{A(T)} \sum_{k=1}^{A(T)} W_k.$$

Assuming there are an infinite number of times  $0 \leq T_i < T_{i+1} < \dots$ ,  $T_i \rightarrow \infty$ , at which  $A(T_i) = D(T_i)$  and the following limits exist

$$\frac{1}{T} \int_0^T L(s) ds \rightarrow E[L], \quad \frac{A(T_i)}{T_i} \rightarrow \lambda, \quad \frac{1}{A(T_i)} \sum_{k=1}^{A(T_i)} W_k \rightarrow E[W],$$

we obtain Little's law.

**s.4.4.4.** Sometimes (often?) students memorize Little's law in the wrong way. Thus, as an easy check, use the dimensions of the concepts:  $E[L]$  is an average *number*,  $\lambda$  is a *rate*, i.e., *numbers per unit time*, and  $E[W]$  is waiting *time*.

**s.4.4.5.** The arrival rate at the server must be  $\lambda$  and the time a job remains at the server is  $E[S]$ . The fraction of time the server is busy is precisely the fraction of time there is a job present at the server. Thus, applying Little's law to the server itself, we see that  $\rho = E[L_S] = \lambda E[S]$ .

**s.4.4.6.**

$$E[W] = E[L]/\lambda = 10/\lambda = 10/5 = 2.$$

**s.4.4.7.** If you arrive at a queueing system, you first have to wait until the job in service is finished. Then you need to wait until the 9 jobs in queue are finished. This takes, in expectation,  $9/\mu$ . (Recall, 1 job is in service at the moment you arrive, so 9 are in queue.) Assuming that service times are exponential, so that, by the memoryless property, the remaining service time of the job in service is still  $E[S]$  when you arrive, you spend  $10/\mu + 1/\mu = 11/6 \neq 2$ . (To account for the last  $+1/\mu$ , observe that yourself also have to be served to compute the time you spend in the system.)

Now in this question, it is *given* that the system length is 10 at the moment of arrival. However,  $L$  as 'seen' upon arrival by this given customer is in general not the same as the time-average  $E[L]$ .

Thus, Little's law need not hold at all moments in time; it is a statement about *averages*.

**s.4.4.8.**

$$\begin{aligned}
E[W] &= E[L] E[S] + E[S] = \lambda E[W] E[S] + E[S] = \rho E[W] + E[S], \\
E[L] &= \lambda E[W] = \frac{\lambda E[S]}{1 - \rho} = \frac{\rho}{1 - \rho}, \\
E[W_Q] &= E[W] - E[S] = \frac{E[S]}{1 - \rho} - E[S] = \frac{\rho}{1 - \rho} E[S], \\
E[L_Q] &= \lambda E[W_Q] = \frac{\rho^2}{1 - \rho}, \\
E[L_s] &= E[L] - E[L_Q] = \frac{\rho}{1 - \rho} - \frac{\rho^2}{1 - \rho} = \rho,
\end{aligned}$$

**s.4.4.9.** By the memoryless property of the (exponential) distributed service times of the  $M/M/1$  queue, the duration of a job in service, if any, is  $\text{Exp}(\mu)$  also at an arrival moment. Therefore, at an arrival moment, all jobs in the system (whether in service or not) have the same expected duration. Hence, the expected time to spend in queue is the expected number of jobs in the system times the expected service time of each job, i.e.,  $E[W_Q] = E[L] E[S]$ . Note that we use PASTA to see that the expected number of jobs in the system at an arrival is  $E[L]$ . For the  $M/G/1$  queue, the job in service (if any) does not have the same distribution as a job in queue. Hence, the expected time in queue is not  $E[L] E[S]$ .

**s.4.4.10.** Intuitively, the left term is all the work that arrived up to time  $t$ , the middle term is all the work that has been processed, and the right term all the work that left. Any job that is half way its service counts for full at the left, for half in the middle expression, and not in the right.

More formally, for any job  $k$  and time  $t$ , we have  $W_k \mathbb{1}_{A_k \leq t} \geq \int_0^t \mathbb{1}_{A_k \leq s < D_k} ds \geq W_k \mathbb{1}_{D_k \leq t}$ . (To see this, fix  $k$ , and check the three cases  $t < A_k, A_k \leq t < D_k, D_k < t$ .) Then,

$$\sum_{k=1}^{\infty} W_k \mathbb{1}_{A_k \leq t} \geq \int_0^t \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq s < D_k} ds \geq \sum_{k=1}^{\infty} W_k \mathbb{1}_{D_k \leq t}.$$

Finally, note that  $\sum_{k=1}^{\infty} W_k \mathbb{1}_{A_k \leq t} = \sum_{k=1}^{A(t)} W_k$  and  $\sum_{k=1}^{\infty} W_k \mathbb{1}_{D_k \leq t} = \sum_{k=1}^{D(t)} W_k$ , and use the definition of  $L(s)$ .

**s.4.4.11.**

$$\lim_{t \rightarrow \infty} \frac{A(t)}{t} \frac{1}{A(t)} \sum_{k=1}^{A(t)} W_k \geq \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t L(s) ds \geq \lim_{t \rightarrow \infty} \frac{D(t)}{t} \frac{1}{D(t)} \sum_{k=1}^{D(t)} W_k.$$

We use the strong law of large numbers to conclude that limits converges to  $n^{-1} \sum_{k=1}^n W_k \rightarrow E[W]$ , and we assume that  $\{W_k, k \geq N\}$  forms a sequence of i.i.d. random variables for  $N$  sufficiently large.

**s.4.4.12.** When  $t > 10$ ,  $L(t) = 10$ . Hence the system is never empty. Since still  $\delta = \lim_{t \rightarrow \infty} D(t)/t = \lambda$ , and  $\lim_{n \rightarrow \infty} n^{-1} \sum_{k=1}^n W_k$  exists, Little's law follows.

**s.4.4.13.** We seem to need that  $\lambda = \delta$  and that  $\lim_{n \rightarrow \infty} n^{-1} \sum_{k=1}^n W_k$  exists.

## 4.5 GRAPHICAL SUMMARY

We finish this chapter with providing two summaries in graphical form to clarify how all concepts developed in this chapter relate.

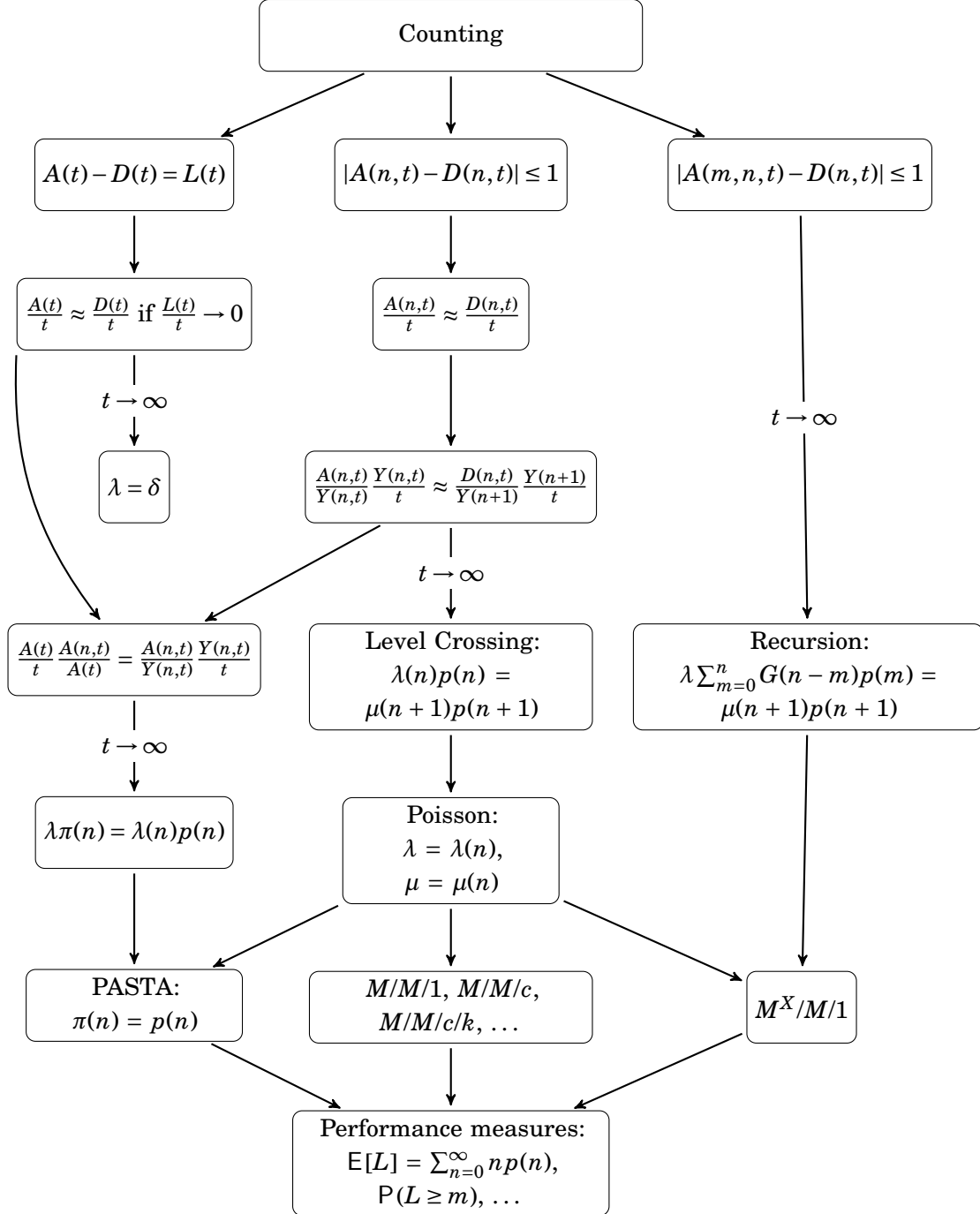


Figure 13: With level-crossing arguments we can derive a number of useful relations. This figure presents an overview of these relations that we derive in this and the next sections.



## EXACT QUEUEING MODELS

In this chapter we use the concepts of Chapter 4 to model and analyze a large number of queueing systems in steady state. The simplest, non-trivial, case is the  $M/M/1$  queue, which is the topic of Section 5.1. As the main ideas are based on sample-paths, it turns out to be nearly trivial to extend the analysis of the  $M/M/1$  to the  $M(n)/M(n)/1$  queue in Section 5.2. The  $M(n)/M(n)/1$  queue is a very generic model with which we can find closed-form expressions for the queue length distributions for many other queueing models such as the  $M/M/c$  queue or the  $M/M/1/K$ , i.e., a queueing system with loss,

We then focus on finding the expected queueing time for batch queues, in Section 5.3, and the  $M/G/1$  queue, in Section 5.4. In the last two sections, Section 5.5 and Section 5.6, we derive expressions for the queue length distributions of the batch queue and the  $M/G/1$  queue.

Many of the queueing systems we analyze here are either generalizations of some other model, for instance, the  $M(n)/M(n)/1$  generalizes to the  $M/M/c$  queue, or reduce to special cases in certain parameter settings, such as that the  $M/G/1$  becomes the  $M/M/1$  queue when the service times are exponentially distributed. Quite a number of exercises in this chapter are targeted on *checking* that the general results reduce to those of the special cases. The reader should understand the importance of such checks. These exercises are simple in a sense—it is perfectly clear what to do, there is no model to make for instance—, but the algebra can be quite tough at times, and hence it is good practice.

### 5.1 $M/M/1$ QUEUE

In the  $M/M/1$  queue, one server serves jobs arriving with exponentially distributed inter-arrival times and each job requires an exponentially distributed processing time. With the level-crossing equations (4.2.7) we derive a number of important results for this queueing process.

#### *Theory and Exercises*

Recall from Section 2.2 that we can construct the  $M/M/1$  queue as a reflected random walk where the arrivals are generated by a Poisson process  $N_\lambda$  and the departures are generated according to the Poisson process  $N_\mu$  (but only during periods at which the system is not empty!). Since the rates of these processes do not depend on the state of the random walk nor on the queue process, it follows that  $\lambda(n) = \lambda$  for all  $n \geq 0$  and  $\mu(n) = \mu$  for all  $n \geq 1$ . Thus, (4.2.7) reduces to

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)} p(n) = \frac{\lambda}{\mu} p(n) = \rho p(n),$$

where we use the definition of the load  $\rho = \lambda/\mu$ . Since this holds for any  $n \geq 0$ , it follows with recursion that

$$p(n+1) = \rho^{n+1} p(0).$$

Then, by using normalization, it follows from (4.2.9) and (0.2.1d) that

$$p(0) = 1 - \rho, \quad p(n) = (1 - \rho)\rho^n. \quad (5.1.1)$$

It is now easy to compute the most important performance measures. The utilization of the server is  $\rho = \lambda/\mu$ , as observed above. Then, with a bit of algebra,

$$E[L] = \frac{\rho}{1-\rho}, \quad V[L] = \frac{\rho}{(1-\rho)^2}, \quad P(L > n) = \rho^{n+1}. \quad (5.1.2)$$

**5.1.1.** What is the interpretation of  $\sum_{n=1}^{\infty} p(n)$ ?

**5.1.2.** Derive the expression for  $E[L]$  in (5.1.2) by means of indicator variables.

**5.1.3.** Show that

$$E[L^2] = (1-\rho) \sum_{n=0}^{\infty} n^2 \rho^n = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho} \quad (5.1.3)$$

by noting that  $\sum_{i=1}^n i = n(n+1)/2$  from which we get that  $n^2 = -n + 2\sum_{i=1}^n i$ . Substitute this relation into  $\sum_n n^2 \rho^n$  and simplify.

**5.1.4.** Derive the expression for  $E[L]$  in (5.1.2) by differentiating the left-hand and right-hand side of the standard formula for a geometric series  $\sum_{n=0}^{\infty} \rho^n = (1-\rho)^{-1}$  for  $\rho < 1$ .

**5.1.5.** Derive (5.1.3) by twice differentiating the standard formula for a geometric series.

**5.1.6.** Use moment-generating functions to derive the expression for  $E[L]$  in (5.1.2).

**5.1.7.** Derive (5.1.3) by using moment-generating functions.

**5.1.8.** Show that for the M/M/1 queue  $V[L] = \rho/(1-\rho)^2$ .

**5.1.9.** Show that the excess probability  $P(L \geq n) = \rho^n$ .

**5.1.10.** Derive (5.1.2) in three ways: by using indicator variables, by means of the expression  $(1-\rho)^{-1} = \sum_{n=0}^{\infty} \rho^n$ , and with moment-generating functions. (This is a good exercise to train your calculus and algebra skills.)

**5.1.11.** Show that for the M/M/1 queue the SCV of  $L$  is  $1/\rho$ . What do you conclude from this?

**5.1.12.** Explain that for the M/M/1 queue  $E[L_Q] = \sum_{n=1}^{\infty} (n-1)\pi(n)$  and use this to find that  $E[L_Q] = \rho^2/(1-\rho)$ .

Let us interpret (5.1.2). The fact that  $E[L] \sim (1-\rho)^{-1}$  for  $\rho \rightarrow 1$  implies that the average waiting time increases very fast when  $\rho \rightarrow 1$ . If we want to avoid long waiting times, this formula tells us that situations with  $\rho \approx 1$  should be avoided. As a practical guideline, it is typically best to keep  $\rho$  quite a bit below 1, and accept that servers are not fully utilized.

Clearly, the probability that the queue length exceeds some threshold decreases geometrically fast (for  $\rho < 1$ ). If we make the simple assumption that customers decide to leave (or rather, not join) the system when the queue is longer than 9 say, then  $P(L \geq 10) = \rho^{10}$  is an estimator for the fraction of customers lost.

**5.1.13 (Hall 5.6).** An M/M/1 queue has been found to have an average waiting time in queue of 1 minute. The arrival rate is known to be 5 customers per minute. What are the service rate and utilization? Calculate  $E[L_Q]$ ,  $E[L]$  and  $E[W]$ . Finally, the queue operator would like to provide chairs for waiting customers. He would like to have a sufficient number so that all customers can sit down at least 90 percent of the time. How many chairs should he provide?

**5.1.14. (Hall 5.5)** An M/M/1 queue has an arrival rate of 100 per hour and a service rate of 140 per hour. What is  $p(n)$ ? What are  $E[L_Q]$  and  $E[L]$ ?



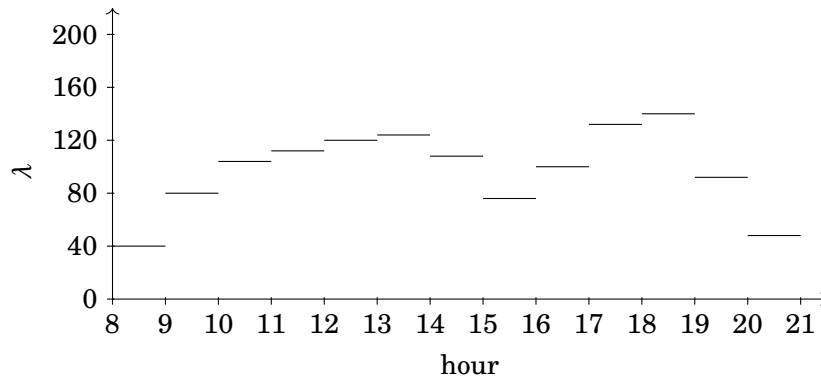


Figure 14: A demand profile of the arrival rate  $\lambda$  modeled as constant over each hour.

**SUPERMARKET PLANNING** Let us consider the example of cashier planning of a supermarket to demonstrate how to use the tools we developed up to now. Out of necessity, our approach is a bit heavy-handed—Turning the example into a practically useful scheme requires more sophisticated queueing models and data assembly—but the present example contains the essential analytic steps to solve the planning problem.

The *service objective* is to determine the minimal service capacity  $c$  (i.e., the number of cashiers) such that the fraction of the time that more than 10 people are in queue is less than 1%. (If the supermarket has 3 cashiers open, 10 people in queue means about 3 people per queue.)

The next step is to find the *relevant data*: the arrival process and the service time distribution. For the arrival process, it is reasonable to model it as a Poisson process. There are many potential customers, each choosing with a small probability to go to the supermarket at a certain moment in time. Thus, we only have to characterize the arrival rate. Estimating this for a supermarket is relatively easy: the cash registers track all customers payments. Thus, we know the number of customers that left the shop, hence entered the shop. (We neglect the time customers spend in the shop.) Based on these data we make a *demand profile*: the average number of customers arriving per hour, cf. Fig. 14. Then we model the arrival process as Poisson with an arrival rate that is constant during a certain hour as specified by the demand profile.

It is also easy to find the service distribution from the cash registers. The first item scanned after a payment determines the start of a new service, and the payment closes the service. (As there is always a bit of time between the payment and the start of a new service we might add 15 seconds, say, to any service.) To keep things simple here, we just model the service time distribution as exponential with a mean of 1.5 minutes.

We also *model* the behavior of all the cashiers together (a multi-server queue) as a single fast server. Thus, we neglect any differences between a station with, for instance, 3 cashiers and a single server that works 3 times as fast as a normal cashier. (We analyze in [5.2.5] the quality of this approximation.) As yet another simplification, we change the objective somewhat such that the number of jobs in the system, rather than the number in queue, should not exceed 10.

We now find a formula to convert the demand profile into the *load profile*, which is the minimal number of servers per hour needed to meet the service objective. We already know for the M/M/1 that  $P(L > 10) = \rho^{11}$ . Combining this with the objective  $P(L > 10) \leq 1\%$ , we get that

$\rho^{11} \leq 0.01$ , which translates into  $\rho \leq 0.67$ . Using that  $\rho = \lambda E[S]/c$  and our estimate  $E[S] = 1.5$  minutes, we get the following rough bound on  $c$ :

$$c \geq \frac{\lambda E[S]}{0.67} \approx \frac{3}{2} \cdot \lambda \cdot 1.5 = 2.25\lambda,$$

where  $\lambda$  is the arrival rate (per minute, *not* per hour). For instance, for the hour from 12 to 13, we read in the demand profile in Fig. 14 that  $\lambda = 120$  customers per hour, hence  $c = 2.25 \cdot 120/60 = 4.5$ . With this formula, the conversion of the demand profile to the load profile becomes trivial: divide the hourly arrival rate by 60 and multiply by 2.25.

The last step is to *cover the load profile with service shifts*. This is typically not easy since shifts have to satisfy all kinds of rules, such as: after 2 hours of work a cashier should take a break of at least 10 minutes; a shift length must be at least four hours, and no longer than 9 hours including breaks; when the shift is longer than 4 hours it needs to contain at least one break of 30 minutes; and so on. These shifts also have different costs: shifts with hours after 18h are more expensive per hour; when the supermarket covers traveling costs, short shifts have higher marginal traveling costs; and so on.

The usual way to solve such covering problems is by means of an integer problem. First, generate all (or a subset of the) allowed shift types with associated starting times. For instance, suppose only 4 shift plans are available

1. ++-++
2. +++-+
3. ++-++++
4. +++-+++,

where a + indicates a working hour and – a break of an hour. Then generate shift types for each of these plans with starting times 8 am, 9 am, and so on, until the end of the day. Thus, a shift type is a shift plan that starts at a certain hour. Let  $x_i$  be the number of shifts of type  $i$  and  $c_i$  the cost of this type. Write  $t \in s_i$  if hour  $t$  is covered by shift type  $i$ . Then the problem is to solve

$$\min \sum_i c_i x_i,$$

such that

$$\sum_i x_i \mathbb{1}_{t \in s_i} \geq 2.25 \frac{\lambda_t}{60}$$

for all hours  $t$  the shop is open and  $\lambda_t$  is the demand for hour  $t$ .

### Hints

**h.5.1.1.** What is the meaning of  $p(0)$ ?

**h.5.1.4.** Observe that we just compute the first moment of a geometric random variable.

**h.5.1.9.**  $P(L \geq n) = \sum_{k \geq n} p(k)$ .

**h.5.1.13.**  $E[L_Q]$  follows right away from an application of Little's law. For the other quantities we need to find  $E[S]$ . One can use that

$$E[W_Q] = E[L] E[S] = (E[L_Q] + E[L_S]) E[S] = (E[L_Q] + \lambda E[S]) E[S].$$

Now  $\lambda$  and  $E[W_Q] = 1$  are given, and  $E[L_Q]$  has just been computed. Hence,  $E[S]$  (which is the unknown here) can be computed with the quadratic formula.

Another way is to realize that, for the M/M/1-queue,  $E[W_Q] = \frac{1}{\lambda} \frac{\rho^2}{1-\rho}$ . Then solve for  $\rho$ , and since  $\lambda$  is known,  $E[S]$  follows.

### Solutions

**s.5.1.1.** First, note that  $p(0)$  must be the fraction of time the server is idle. Hence, the fraction of time the server is busy, i.e., the utilization, is

$$1 - p(0) = \rho = \sum_{n=1}^{\infty} p(n).$$

Here the last equation has the interpretation of the fraction of time the system contains at least 1 job.

**s.5.1.2.** A bit long, but I spell out every step:

$$\begin{aligned} E[L] &= \sum_{n=0}^{\infty} n p(n) = \sum_{n=0}^{\infty} \sum_{i=1}^n \mathbb{1}_{i \leq n} p(n) & n &= \sum_{i=1}^n \mathbb{1}_{i \leq n} \\ &= \sum_{n=0}^{\infty} \sum_{i=1}^{\infty} \mathbb{1}_{i \leq n} p(n) & i > n &\implies \mathbb{1}_{i \leq n} = 0 \\ &= \sum_{i=1}^{\infty} \sum_{n=0}^{\infty} \mathbb{1}_{i \leq n} p(n) = \sum_{i=1}^{\infty} \sum_{n=i}^{\infty} p(n) & n < i &\implies \mathbb{1}_{i \leq n} = 0 \\ &= \sum_{i=1}^{\infty} \sum_{n=i}^{\infty} (1-\rho) \rho^n & p(n) &= (1-\rho) \rho^n \\ &= \sum_{i=1}^{\infty} \sum_{n=0}^{\infty} (1-\rho) \rho^{n+i} & n &\rightarrow n+i \\ &= \sum_{i=1}^{\infty} (1-\rho) \rho^i \sum_{n=0}^{\infty} \rho^n & \rho^{n+i} &= \rho^i \rho^n \\ &= \sum_{i=1}^{\infty} (1-\rho) \rho^i \frac{1}{1-\rho} = \sum_{i=1}^{\infty} \rho^i = \sum_{i=0}^{\infty} \rho^{i+1} & i &\rightarrow i+1 \\ &= \rho \sum_{i=0}^{\infty} \rho^i = \frac{\rho}{1-\rho}. \end{aligned}$$

Note that, since the summands are positive, we can use Fubini's theorem to justify the interchange of the summations.

**s.5.1.3.**

$$\begin{aligned} \sum_{n=0}^{\infty} n^2 \rho^n &= \sum_{n=0}^{\infty} \left( \sum_{i=1}^{\infty} 2i \mathbb{1}_{i \leq n} - n \right) \rho^n = \sum_{n=0}^{\infty} \sum_{i=0}^{\infty} 2i \mathbb{1}_{i \leq n} \rho^n - \sum_{n=0}^{\infty} n \rho^n = \sum_{i=0}^{\infty} 2i \sum_{n=i}^{\infty} \rho^n - \frac{E[L]}{1-\rho} \\ &= \sum_{i=0}^{\infty} 2i \rho^i \sum_{n=0}^{\infty} \rho^n - \frac{E[L]}{1-\rho} = \frac{2}{1-\rho} \sum_{i=0}^{\infty} i \rho^i - \frac{E[L]}{1-\rho} = \frac{2}{(1-\rho)^2} E[L] - \frac{E[L]}{1-\rho} \end{aligned}$$

$$\begin{aligned}
&= \frac{E[L]}{1-\rho} \left( \frac{2}{1-\rho} - 1 \right) = \frac{E[L]}{1-\rho} \frac{1+\rho}{1-\rho} \\
&= \frac{\rho}{1-\rho} \frac{1+\rho}{(1-\rho)^2}.
\end{aligned}$$

**s.5.1.4.** Differentiate the left- and right-hand side of  $(1-\rho)^{-1} = \sum_{n=0}^{\infty} \rho^n$  with respect to  $\rho$  and then multiply with  $\rho$  to get

$$\frac{\rho}{(1-\rho)^2} = \sum_{n=0}^{\infty} n \rho^n.$$

Then multiply both sides by  $1-\rho$  and use that  $p(n) = (1-\rho)\rho^n$ .

**s.5.1.5.** Starting from the result of [5.1.4], differentiating and multiplying with  $\rho$  a second time yields

$$\begin{aligned}
\rho \frac{(1-\rho)^2 + \rho 2(1-\rho)}{(1-\rho)^4} &= \rho \frac{1-2\rho+\rho^2+2\rho-2\rho^2}{(1-\rho)^4} = \rho \frac{1-\rho^2}{(1-\rho)^4} \\
&= \rho \frac{1+\rho}{(1-\rho)^3} = \sum_{n=0}^{\infty} n^2 \rho^n,
\end{aligned}$$

and hence

$$\begin{aligned}
(1-\rho) \sum_{n=0}^{\infty} n^2 \rho^n &= \rho \frac{1+\rho}{(1-\rho)^2} = \frac{\rho}{(1-\rho)^2} + \frac{\rho^2}{(1-\rho)^2} = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} \\
&= \frac{2\rho^2}{(1-\rho)^2} + \rho \frac{(1-\rho)}{(1-\rho)^2} = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{(1-\rho)^2}.
\end{aligned}$$

Recall that  $p(n) = (1-\rho)\rho^n$ .

**s.5.1.6.**

$$M_L(s) = E[e^{sL}] = \sum_{n=0}^{\infty} e^{sn} p(n) = (1-\rho) \sum_{n=0}^{\infty} e^{sn} \rho^n = \frac{1-\rho}{1-e^s \rho},$$

where we assume that  $s$  is such that  $e^s \rho < 1$ . Then,

$$M'_L(s) = (1-\rho) \frac{1}{(1-e^s \rho)^2} e^s \rho.$$

Hence,  $E[L] = M'_L(0) = \rho/(1-\rho)$ .

**s.5.1.7.** Using the results of [5.1.6] gives

$$E[L^2] = M''(0) = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho}.$$

**s.5.1.8.**

$$V[L] = E[L^2] - (E[L])^2 = \frac{\rho(1+\rho)}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} = \frac{\rho}{(1-\rho)^2}.$$

**s.5.1.9.**

$$\begin{aligned}
P(L \geq n) &= \sum_{k=n}^{\infty} p(k) = \sum_{k=n}^{\infty} p(0) \rho^k = (1-\rho) \sum_{k=n}^{\infty} \rho^k \\
&= (1-\rho) \rho^n \sum_{k=0}^{\infty} \rho^k = (1-\rho) \rho^n \frac{1}{1-\rho} = \rho^n.
\end{aligned}$$

**s.5.1.10.** See [5.1.2] to [5.1.9].

**s.5.1.11.** To see how large this variance is, relative to the mean number of jobs in the system, we typically consider the squared coefficient of variation (SCV). As  $E[L] = \rho/(1 - \rho)$ ,

$$\frac{V[L]}{(E[L])^2} = \frac{1}{\rho}.$$

Thus, the SCV becomes smaller as  $\rho$  increases, but does not become lower than 1. So, realizing that the SCV of the exponential distribution is 1, the distribution of the number of jobs in the system has larger relative variability than the exponential distribution.

**s.5.1.12.** The fraction of time the system contains  $n$  jobs is  $\pi(n)$  (by PASTA). When the system contains  $n > 0$  jobs, the number in queue is one less, i.e.,  $n - 1$ .

$$\begin{aligned} E[L_Q] &= \sum_{n=1}^{\infty} (n-1)\pi(n) = (1-\rho) \sum_{n=1}^{\infty} (n-1)\rho^n \\ &= \rho(1-\rho) \sum_{n=1}^{\infty} (n-1)\rho^{n-1} = \rho \sum_{n=1}^{\infty} (n-1)\pi(n-1) \\ &= \rho \sum_{n=0}^{\infty} n\pi(n) = \rho \frac{\rho}{1-\rho}. \end{aligned}$$

Another way to get the same result is by splitting:

$$\begin{aligned} E[L_Q] &= \sum_{n=1}^{\infty} (n-1)\pi(n) = \sum_{n=1}^{\infty} n\pi(n) - \sum_{n=1}^{\infty} \pi(n) \\ &= E[L] - (1 - \pi(0)) = E[L] - \rho. \end{aligned}$$

**s.5.1.13.** Note that  $E[W_Q] = \frac{1}{\lambda} \frac{\rho^2}{1-\rho}$ . Since  $E[W_Q]$  and  $\lambda$  are given, we can use this formula to solve for  $\rho$  with the quadratic formula (and using that  $\rho > 0$ ):

```
>>> from math import sqrt
>>> labda = 5. # per minute
>>> Wq = 1.
>>> a = 1.
>>> b = labda*Wq
>>> c = -labda*Wq
>>> rho = (-b + sqrt(b*b-4*a*c))/(2*a)
>>> rho
0.8541019662496847

>>> ES = rho/labda
>>> ES
0.17082039324993695

>>> Lq = labda*Wq
>>> Lq
5.0
```

```

>>> W = Wq + ES
>>> W
1.170820393249937

>>> L = labda*W
>>> L
5.854101966249685

```

The next problem is to find  $n$  such that  $\sum_{j=0}^n p_j > 0.9$ .

```

>>> total = 0.
>>> j = 0
>>> while total <= 0.9:
...     total += (1-rho)*rho**j
...     j += 1
...
>>> total
0.9061042738330157
>>> j
15
>>> n = j - 1 # the number of chairs

```

Observe that  $j$  is one too high once the condition is satisfied, thus subtract one.  
As a check, I use that  $(1 - \rho) \sum_{j=0}^n \rho^j = 1 - \rho^{n+1}$ .

```

>>> 1-rho**(n) # this must be too small.
0.890064968964683
>>> 1-rho**(n+1) # this must be OK.
0.9061042738330156

```

And indeed, we found the right  $n$ .

**s.5.1.14.** First,  $p(n) = (1 - \rho)\rho^n$ . Now,

```

>>> labda = 100. # per hour
>>> mu = 140. # per hour
>>> ES = 1./mu
>>> rho = labda/mu
>>> rho
0.7142857142857143
>>> 1-rho
0.2857142857142857

>>> L = rho/(1.-rho)
>>> L
2.5
>>> Lq = rho**2/(1.-rho)

```

```

>>> Lq
1.7857142857142858

>>> W = 1./(1.-rho) * ES
>>> W
0.024999999999999998
>>> Wq = rho/(1.-rho) * ES
>>> Wq
0.017857142857142856

```

## 5.2 $M(n)/M(n)/1$ QUEUE

As it turns out, many more single-server queueing situations than the  $M/M/1$  queue can be analyzed by making a judicious choice of  $\lambda(n)$  and  $\mu(n)$  in the level-crossing equations (4.2.7). For these queueing systems, we just present the results. The first set of exercises are somewhat technical; we ask you to derive the formulas—the main challenge is not to make computational errors. The second set of exercises shows how the combination of PASTA and Little's law allows us to analyze an astonishingly large number of non-trivial practical queueing situations.

### *Theory and Exercises*

It is important to realize that the inter-arrival times and service times need to be memoryless for the analysis below; the rates, however, may depend on the number of jobs in the system. Specifically, we require that for all  $s$  and  $t$ ,

$$P(A_{A(t)+1} \leq t+s \mid L(t) = n) = 1 - e^{-\lambda(n)s},$$

where we use that  $A_{A(t)+1}$  is the arrival time of the next job after time  $t$ . Similarly, we assume for all  $t$  and  $s$ ,

$$P(D_{D(t)+1} \leq t+s \mid L(t) = n) = 1 - e^{-\mu(n)s}.$$

**5.2.1.** Model the  $M/M/1/K$  queue in terms of an  $M(n)/M(n)/1$  queue and compute  $p(K)$ , i.e., the fraction of time that the system is full.

**5.2.2.** Show that as  $K \rightarrow \infty$ , the performance measures of the  $M/M/1/K$  converge to those of the  $M/M/1$  queue.

**5.2.3.** Model the  $M/M/c$  queue in terms of an  $M(n)/M(n)/1$  queue and compute  $E[L_Q]$ .

**5.2.4.** Check that the performance measures of the  $M/M/c$  queue reduce to those of the  $M/M/1$  queue if  $c = 1$ .

**5.2.5.** It should be clear that the  $M/M/c$  queue is a bit harder to analyze than the  $M/M/1$  queue, at least the expressions are more extensive. It is tempting to approximate the  $M/M/c$  queue by an  $M/M/1$  queue with a server that works  $c$  times as fast. As we now have the formulas for the  $M/M/c$  queue and the  $M/M/1$  queue we can use these to obtain some basic understanding of the difference.

Let us therefore consider a numerical example. Suppose that we have an  $M/M/3$  queue, with arrival rate  $\lambda = 5$  per day and  $\mu = 2$  per server, and we compare it to an  $M/M/1$  with the same

arrival rate but with a service rate of  $\mu = 3 \cdot 2 = 6$ . Make a graph of the ratios of  $E[L]$  and  $E[L_Q]$  of both models as a function of  $\rho$ . Explain why these ratios become 1 as  $\rho \uparrow 1$ .

**5.2.6.** Model the  $M/M/c/c$  queue in terms of an  $M(n)/M(n)/1$  queue and determine the performance measures. This model is also known as the Erlang B-formula and is often used to determine the number of beds at hospitals, where the beds act as servers and the patients as jobs.

**5.2.7.** Take the limit  $c \rightarrow \infty$  in the  $M/M/c$  queue (or the  $M/M/c/c$  queue) and obtain the performance measures for the  $M/M/\infty$  queue, i.e., a queueing system with ample servers.

**5.2.8.** Show that the  $M/M/\infty$  queue is stable for any finite  $\lambda$ .

**5.2.9.** Why is  $E[L] = \rho$  for the  $M/M/\infty$  queue?

**5.2.10.** Consider the  $M/M/2/3$  queue with arrival rate  $\lambda$  and service rate  $\mu$  (thus, at most 2 jobs can be in service and 1 in queue). Derive first the level-crossing equations for this queueing system, then derive closed form expressions for the state probabilities in steady state.

**5.2.11** (Multi-server queue with blocking). Consider the  $M/M/c/c + K$  queue in which at most  $c$  jobs can be in service and  $K$  in queue. Try to derive the steady-state probabilities  $p(0), p(1), \dots$ . You do not have to compute the normalization constant  $G$ .

**5.2.12.** Derive the steady state probabilities  $p(n)$  for a single-server queue with a finite calling population with  $N$  jobs, i.e., jobs that are in service cannot arrive to the system. Check the answer you obtained for the cases  $N = 1$  and  $N = 2$ . What happens if  $N \rightarrow \infty$ ? Interpret the results.

**5.2.13.** Give an example of a system with a finite calling population.

**5.2.14.** Derive the steady state probabilities  $p(n)$  for a queue with a finite calling population with  $N$  jobs and  $N$  servers, i.e., the number of servers in the queueing system is equal to the size of the calling population. What happens if  $N \rightarrow \infty$ ?

Finally, we consider queues with *balking*, that is, queues in which customers leave when they find the queue too long at the moment they arrive. A simple example model with customer balking is given by

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n = 0, \\ \lambda/2, & \text{if } n = 1, \\ \lambda/4, & \text{if } n = 2, \\ 0, & \text{if } n > 2, \end{cases}$$

and  $\mu(n) = \mu$ .

Observe that here we make a subtle implicit assumption; in Section 4.3 we elaborate on this assumption. To make the problem clear, note that balking customers *decide at the moment they arrive* to either join or leave; in other words, they decide based on what they ‘see upon arrival’. In yet other words, they make decisions based on the state of the system at arrival moments, not on time-averages. However, the notion of  $p(n)$  is a long-run *time-average*, and is typically not the same as what customers ‘see upon arrival’. As a consequence, the performance measure  $P(L \leq n)$  is not necessarily in accordance with the perception of customers. To relate these two ‘views’, i.e., time-average versus observer-average, we need a new concept, *PASTA*, to be developed in Section 4.3.



**5.2.15.** In what way is a queueing system with balking, at level  $b$  say, different from a queueing system with finite calling population of size  $b$ ?

**5.2.16** (Hall 5.2). After observing a single-server queue for several days, the following steady-state probabilities have been determined:  $p(0) = 0.4$ ,  $p(1) = 0.3$ ,  $p(2) = 0.2$ ,  $p(3) = 0.05$  and  $p(4) = 0.05$ . The arrival rate is 10 customers per hour.

1. Determine  $E[L]$  and  $E[L_Q]$ .
2. Using Little's formula, determine  $E[W]$  and  $E[W_Q]$ .
3. Determine  $V[L]$  and  $V[L_Q]$ .
4. Determine the service time and the utilization.

**5.2.17.** (Hall 5.7) A single-server queueing system is known to have Poisson arrivals and exponential service times. However, the arrival rate and service time are state dependent. As the queue becomes longer, servers work faster, and the arrival rate declines, yielding the following functions (all in units of number per hour):  $\lambda(0) = 5$ ,  $\lambda(1) = 3$ ,  $\lambda(2) = 2$ ,  $\lambda(n) = 0, n \geq 3$ ,  $\mu(0) = 0$ ,  $\mu(1) = 2$ ,  $\mu(2) = 3$ ,  $\mu(n) = 4, n \geq 3$ . Calculate the state probabilities, i.e.,  $p(n)$  for  $n = 0, \dots$

Why  $p(n) \neq (1 - \rho)\rho^n$ ?

**5.2.18.** (Hall 5.14) An airline phone reservation line has one server and a buffer for two customers. The arrival rate is 6 customers per hour, and a service rate of just 5 customers per hour. Arrivals are Poisson and service times are exponential. Estimate  $E[L_Q]$  and the average number of customers served per hour. Then, estimate  $E[L_Q]$  for a buffer of size 5. What is the impact of the increased buffer size on the number of customers served per hour?

**5.2.19** (Hall 5.3). After observing a queue with two servers for several days, the following steady-state probabilities have been determined:  $p(0) = 0.4$ ,  $p(1) = 0.3$ ,  $p(2) = 0.2$ ,  $p(3) = 0.05$  and  $p(4) = 0.05$ . The arrival rate is 10 customers per hour.

1. Determine  $E[L]$  and  $E[L_Q]$ .
2. Using Little's formula, determine  $E[W]$  and  $E[W_Q]$ .
3. Determine  $V[L]$  and  $V[L_Q]$ .
4. Determine the service time and the utilization.

**5.2.20** (Hall 5.8). The queueing system at a fast-food stand behaves in a peculiar fashion. When there is no one in the queue, people are reluctant to use the stand, fearing that the food is unsavory. People are also reluctant to use the stand when the queue is long. This yields the following arrival rates (in numbers per hour):  $\lambda(0) = 10$ ,  $\lambda(1) = 15$ ,  $\lambda(2) = 15$ ,  $\lambda(3) = 10$ ,  $\lambda(4) = 5$ ,  $\lambda(n) = 0, n \geq 5$ . The stand has two servers, each of which can operate at 5 per hour. Service times are exponential, and the arrival process is Poisson. Calculate the steady state probabilities. Next, what is the average arrival rate? Finally, determine  $E[L]$ ,  $E[L_Q]$ ,  $E[W]$  and  $E[W_Q]$ .

**5.2.21** (Hall 5.10). A repair/maintenance facility would like to determine how many employees should be working in its tool crib. The service time is exponential, with mean 4 minutes, and customers arrive by a Poisson process with rate 28 per hour. The customers are actually maintenance workers at the facility, and are compensated at the same rate as the tool crib employees. What is  $E[W]$  for  $c = 1, 2, 3$ , or 4 servers? How many employees should work in the tool crib?

**5.2.22** (Hall 5.22). *At a large hotel, taxi cabs arrive at a rate of 15 per hour, and parties of riders arrive at the rate of 12 per hour. Whenever taxicabs are waiting, riders are served immediately upon arrival. Whenever riders are waiting, taxicabs are loaded immediately upon arrival. A maximum of three cabs can wait at a time (other cabs must go elsewhere).*

1. *Let  $p_{ij}$  be the steady-state probability of there being  $i$  parties of riders and  $j$  taxicabs waiting at the hotel. Write the state transition equation for the system.*
2. *Calculate the expected number of cabs waiting and the expected number of parties waiting.*
3. *Calculate the expected waiting time for cabs and the expected waiting time for parties. (For cabs, compute the average among those that do not go elsewhere.)*
4. *In words, what would be the impact of allowing four cabs to wait at a time?*

**5.2.23** (Continuation of [5.2.22]). *Suppose cabs are not allowed to wait. What is the expected waiting time for a party of riders?*

**5.2.24** (Continuation of [5.2.22]). *Did you have to use the PASTA property to solve [5.2.22]? If so, how did you use it? If not, why not?*

**5.2.25** (Continuation of [5.2.22]). *Suppose cabs can contain at most 4 riders, and the size of a party (i.e., a batch) has distribution  $B_k$  with  $P(B_k = i) = 1/7$  for  $i = 1, \dots, 7$ . Parties of riders have the same destination, so riders of different parties cannot be served by one taxi. Provide a set of recursions to simulate this system. (This is a real hard exercise, but doable. I asked it at an exam to see who would deserve the highest grade. I was lenient with the grading...)*

### Hints

**h.5.2.1.** Take  $\lambda(n) = \lambda \mathbb{1}_{n < K}$ ,  $\mu(n) = \mu$ , use the equations around (4.2.8).

**h.5.2.2.** Use that  $\sum_{i=0}^n x^i = (1 - x^{n+1})/(1 - x)$ . BTW, is it necessary for this expression to be true that  $|x| < 1$ ? What should you require for  $|x|$  when you want to take the limit  $n \rightarrow \infty$ ?

**h.5.2.3.** Take  $\lambda(n) = \lambda$ , and  $\mu(n) = \min\{n, c\}\mu$ .

**h.5.2.4.** Fill in  $c = 1$ . Realize that this is a check on the formulas.

**h.5.2.7.** Use that for any  $x$ ,  $x^n/n! \rightarrow 0$  as  $n \rightarrow \infty$ .

**h.5.2.10.** Think about what would be the appropriate model choices for  $\lambda(n)$  and  $\mu(n)$  and use the level-crossing equations  $\lambda(n)p(n) = \mu(n+1)p(n+1)$ . For instance, realize that  $\lambda(3) = 0$ : the system cannot contain more than 3 jobs, hence a state with 4 jobs must be impossible. We can achieve that by setting  $\lambda(3) = 0$ . For the service rate, how many servers are busy when the system contains 2 or more jobs? What does this say about  $\mu(k)$  for  $k = 2$  or  $k = 3$ .

**h.5.2.11.** Use  $\lambda(n)p(n) = \mu(n+1)p(n+1)$  and find suitable expressions for  $\lambda(n)$  and  $\mu(n+1)$ .

**h.5.2.12.** Use  $\lambda(n)p(n) = \mu(n+1)p(n+1)$ , and realize that for this case  $\lambda(n) = (N - n)\lambda$  and  $\mu(n) = \mu$ .

**h.5.2.17.** Use the level-crossing equations of the  $M(n)/M(n)/1$  queue.

**h.5.2.18.** This is a queueing system with loss, in particular the  $M/M/1/1 + 2$  queue.

**h.5.2.19.** Solve [5.2.16] first.

**h.5.2.21.** Realize that we have to control the number of servers. Hence, we are dealing with a multi-server queue, i.e., the  $M/M/c$  queue. Use [5.2.3].

The remark that maintenance workers are compensated at the same rate as the tool crib workers confused me a bit at first. Some thought revealed that the consequence of this remark is that it is just as expensive to let the tool crib workers wait (to help maintenance workers) as to let the maintenance workers wait for tools. (Recall, in queueing systems always somebody has to wait, either the customer in queue or the server being idle. If it is very expensive to let customers wait, the number of servers must be high, whereas if servers are relatively expensive, customers have to do the waiting.)

**h.5.2.25.** Realize that you have model the server process separately from the queueing process.

### Solutions

**s.5.2.1.** Note that

$$1 = \sum_{i=0}^K p(i) = p(0) \sum_{i=0}^K \rho^i = p(0) \frac{1 - \rho^{K+1}}{1 - \rho}.$$

Thus,

$$p(n) = \frac{\rho^n}{G}, \quad 0 \leq n \leq K, \quad (5.2.1a)$$

$$p(K) = \frac{1 - \rho}{1 - \rho^{K+1}} \rho^K. \quad (5.2.1b)$$

**s.5.2.2.** To take the limit  $K \rightarrow \infty$ —mind, not the limit  $n \rightarrow \infty$ —, write

$$G = \frac{1 - \rho^{K+1}}{1 - \rho} = \frac{1}{1 - \rho} - \frac{\rho^{K+1}}{1 - \rho}.$$

Since  $\rho^{K+1} \rightarrow 0$  as  $K \rightarrow \infty$  (recall,  $\rho < 1$ ), we get

$$G \rightarrow \frac{1}{1 - \rho},$$

as  $K \rightarrow \infty$ . Therefore,  $p(n) = \rho^n/G \rightarrow \rho^n(1 - \rho)$ , and the latter are the steady-state probabilities of the  $M/M/1$  queue. Finally, if the steady-state probabilities are the same, the performance measures (which are derived from  $p(n)$ ) must be the same.

**s.5.2.3.** First we use the hint to establish a generic relation for  $p(n)$ . Taking  $\rho = \lambda/(c\mu)$ ,

$$\begin{aligned} p(n) &= \frac{\lambda(n-1)}{\mu(n)} p(n-1) = \frac{\lambda}{\min\{c, n\}\mu} p(n-1) = \frac{1}{\min\{c, n\}} (c\rho) p(n-1) \\ &= \frac{1}{\min\{c, n\} \min\{c, n-1\}} (c\rho)^2 p(n-2) \end{aligned}$$

$$= \frac{1}{\prod_{k=1}^n \min\{c, k\}} (c\rho)^n p(0).$$

Thus, if  $n < c$ :

$$p(n) = \frac{(c\rho)^n}{n!} p(0). \quad (5.2.2)$$

If  $n \geq c$ :

$$\begin{aligned} p(n) &= \frac{1}{\prod_{k=1}^c k \cdot \prod_{k=c+1}^n c} (c\rho)^n p(0) \\ &= \frac{1}{c! c^{n-c}} c^n \rho^n p(0) \\ &= \frac{c^c}{c!} \rho^n p(0). \end{aligned}$$

To obtain the normalization constant  $G$ ,

$$\begin{aligned} 1 &= \sum_{n=0}^{\infty} p(n) = \sum_{n=0}^{c-1} p(n) + \sum_{n=c}^{\infty} p(n) \\ &= p(0) \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + p(0) \sum_{n=c}^{\infty} \frac{c^c}{c!} \rho^n \\ &= p(0) \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + p(0) \sum_{n=c}^{\infty} \frac{(c\rho)^c}{c!} \rho^{n-c} \\ &= p(0) \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + p(0) \frac{(c\rho)^c}{c!} \sum_{n=0}^{\infty} \rho^n \\ &= p(0) \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + p(0) \frac{(c\rho)^c}{c!(1-\rho)}. \end{aligned}$$

Hence,

$$p(0)^{-1} = G = \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}. \quad (5.2.3)$$

Next,

$$\begin{aligned} E[L_Q] &= \sum_{n=c}^{\infty} (n-c)p(n) \\ &= \sum_{n=c}^{\infty} (n-c) \frac{c^c}{c!} \rho^n p(0) \\ &= \frac{c^c \rho^c}{Gc!} \sum_{n=c}^{\infty} (n-c) \rho^{n-c} \\ &= \frac{c^c \rho^c}{Gc!} \sum_{n=0}^{\infty} n \rho^n = \frac{c^c \rho^c}{Gc!} \frac{\rho}{(1-\rho)^2}. \end{aligned}$$

The derivation of the expected number of jobs in service becomes easier if we pre-multiply the normalization constant  $G$ :

$$\begin{aligned} G E[L_S] &= G \left( \sum_{n=0}^c n p(n) + \sum_{n=c+1}^{\infty} c p(n) \right) \\ &= \sum_{n=1}^c n \frac{(c\rho)^n}{n!} + \sum_{n=c+1}^{\infty} c \frac{c^c \rho^n}{c!} = \sum_{n=1}^c \frac{(c\rho)^n}{(n-1)!} + \frac{c^{c+1}}{c!} \sum_{n=c+1}^{\infty} \rho^n \end{aligned}$$

$$= \sum_{n=0}^{c-1} \frac{(c\rho)^{n+1}}{n!} + \frac{(c\rho)^{c+1}}{c!} \sum_{n=0}^{\infty} \rho^n = c\rho \left( \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)} \right).$$

Observe that the right-hand side is precisely equal to  $\rho cG$ , and hence,

$$E[L_S] = c\rho = \frac{\lambda}{\mu}.$$

**s.5.2.4.** Take  $c = 1$

$$p(n) = \frac{1}{G} \frac{(c\rho)^0}{0!} = \frac{1}{G}, \quad n = 0, \dots, 1-1 \quad (5.2.4a)$$

$$p(n) = \frac{1}{G} \frac{c^c \rho^n}{c!} = \frac{1}{G} \frac{1^1 \rho^n}{1!} = \frac{\rho^n}{G}, \quad n = 1, 1+1, \dots \quad (5.2.4b)$$

$$G = \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} = \sum_{n=0}^0 \frac{\rho^0}{0!} + \frac{\rho}{(1-\rho)} = 1 + \frac{\rho}{1-\rho} = \frac{1}{1-\rho}, \quad (5.2.4c)$$

$$E[L_Q] = \frac{(c\rho)^c}{c!G} \frac{\rho}{(1-\rho)^2} = \frac{\rho}{1/(1-\rho)} \frac{\rho}{(1-\rho)^2} = \frac{\rho^2}{1-\rho}, \quad (5.2.4d)$$

$$E[L_S] = \sum_{n=0}^c n p(n) + \sum_{n=c+1}^{\infty} c p(n) = p(1) + 1 \sum_{n=2}^{\infty} p(n) = 1 - p(0) = \rho. \quad (5.2.4e)$$

Everything is in accordance to the formulas we derived earlier for the  $M/M/1$  queue.

**s.5.2.5.** I implement the formulas of [5.2.3] in Python. First the results for the  $M/M/3$  queue.

```
>>> from math import exp, factorial

>>> labda = 5
>>> mu = 2
>>> c = 3

>>> rho = labda / mu / c
>>> rho
0.8333333333333334

>>> G = sum((c * rho) ** n / factorial(n) for n in range(c))
>>> G += (c * rho) ** c / ((1 - rho) * factorial(c))
>>> G
22.250000000000004

>>> ELQ = (c * rho) ** c / (factorial(c) * G) * rho / (1 - rho) ** 2
>>> ELQ
3.511235955056181
>>> ELS = rho * c
>>> ELS
2.5
>>> EL = ELQ + ELS
>>> EL
6.011235955056181
```

Now for the  $M/M/1$  queue:

```
>>> labda = 5
>>> c = 3
>>> mu = 2 * c

>>> rho = labda / mu
>>> rho
0.8333333333333334

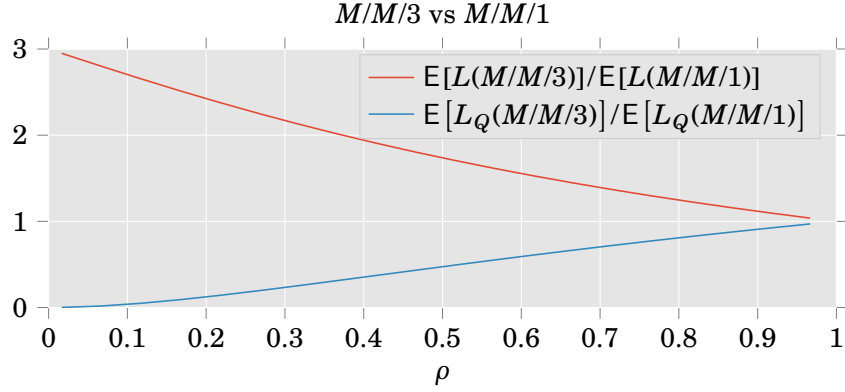
>>> ELQ = rho ** 2 / (1 - rho)
>>> ELQ
4.166666666666667
>>> ELS = rho
>>> ELS
0.8333333333333334
>>> EL = ELS + ELQ
>>> EL
5.000000000000001

>>> rho / (1 - rho) # this must also be EL, just a check
5.000000000000002
```

Note the last check. As a rule, you should always compare your results with known results. BTW, that is one of the reasons I prefer to code the formulas instead of using a calculator. Testing with code is relatively easy, whereas with a calculator it is impossible. (You simply can't check what you typed at the calculator.)

So, returning to the results, as expected, the number of jobs in queue is smaller for the  $M/M/3$  queue, but the number in service is higher.

To put things in a larger perspective, see the figure below where we plot the ratio of the queue lengths and the system length as functions of  $\rho$ . We see, in case of high load, that  $E[L_Q]$  and  $E[L]$  are nearly the same for both systems. This is as expected: when the load is high, most jobs should be in the queue. Therefore,  $E[L_Q]/E[L] \rightarrow 1$  as  $\rho \rightarrow 1$ . When  $\rho$  is small, the difference is quite large. This is also reasonable, because the service time in the fast  $M/M/1$  is 3 times as small as the service time in the  $M/M/3$  queue. Hence, as  $\rho$  is small, the time in the system is dominated by service time, as there is hardly any queueing time, if at all. Thus, there must be more jobs in the system on average in the  $M/M/3$  queue than in the fast  $M/M/1$  queue.



The code can be found on github in the progs directory.

**s.5.2.6.** Take,  $\lambda(n) = \lambda$  if  $n < c$ , and  $\lambda(n) = 0$  for  $n \geq c$ . Also, let,  $\mu(n) = n\mu$  for  $n \leq c$ . (And  $n$  can never be larger than  $c$ , since  $\lambda(n) = 0$  for  $n \geq c$ .) Define  $\rho = \lambda/(c\mu)$ . Then, we see that  $p(n) = p(0)(c\rho)^n/n!$ . For the normalization

$$1 = \sum_{n=0}^c p(n) = p(0) \sum_{n=0}^c \frac{(c\rho)^n}{n!}.$$

Thus, the normalization constant  $G = \sum_{n=0}^c \frac{(c\rho)^n}{n!}$ , and  $p(0) = G^{-1}$ .

Since there are as many servers as places available in the system,  $E[L_Q] = 0$ . The expected number of servers busy is

$$\begin{aligned} E[L_S] &= \sum_{n=0}^c n p(n) = \sum_{n=1}^c n p(n) \\ &= G^{-1} \sum_{n=1}^c n \frac{(\lambda/\mu)^n}{n!} = G^{-1} \sum_{n=1}^c \frac{(\lambda/\mu)^n}{(n-1)!} \\ &= \frac{\lambda}{\mu G} \sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} = \frac{\lambda}{G\mu} \left( G - \frac{(\lambda/\mu)^c}{c!} \right) \\ &= \frac{\lambda}{\mu} \left( 1 - \frac{1}{G} \frac{(\lambda/\mu)^c}{c!} \right) \\ &= \frac{\lambda}{\mu} (1 - p(c)). \end{aligned}$$

This can be explained as follows:  $\lambda(1 - p(c))$  is the rate of accepted jobs (since a fraction  $p(c)$  is lost). Thus, the load is  $\lambda(1 - p(c))/\mu$ , and the load is the fraction of time the servers are busy.

**s.5.2.7.** By taking the limit  $c \rightarrow \infty$ , note first that in (5.2.3),

$$\frac{(c\rho)^c}{(1-\rho)c!} = \frac{(\lambda/\mu)^c}{(1-\rho)c!} \rightarrow 0, \quad \text{as } c \rightarrow \infty.$$

Hence

$$G = \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} \rightarrow \sum_{n=0}^{\infty} \frac{(c\rho)^n}{n!} = e^{\lambda/\mu}.$$

Next, for any fixed  $n$ , eventually  $c > n$ , and then, as  $\rho = \lambda/(\mu c)$ ,

$$p(n) = \frac{1}{G} \frac{(c\rho)^n}{n!} = \frac{1}{G} \frac{(\lambda/\mu)^n}{n!} \rightarrow e^{-\lambda/\mu} \frac{(\lambda/\mu)^n}{n!}, \quad \text{as } c \rightarrow \infty.$$

Moreover, there is no fixed  $n$  such that  $n > c$ . Thus, the probabilities in [5.2.3] are no longer present. Thus, we see that the number of busy servers in the  $M/M/\infty$  queue is Poisson distributed with parameter  $\lambda/\mu$ , and  $E[L] = E[L_S] = \lambda/\mu$ . Observe that now  $\lambda/\mu$  has no longer the interpretation of the fraction of time the server(s) are busy; it is the average number of busy servers.

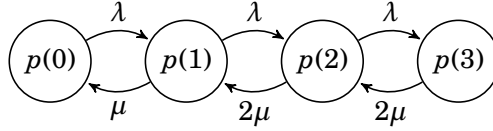
We mention in passing—but do not prove it—that the same results also hold for the  $M/G/\infty$  queue with  $\lambda E[S]$  rather than  $\lambda/\mu$ .

**s.5.2.8.** No matter how many jobs are in service, there is always another free server available when a new job arrives. Thus, jobs never have to wait in queue, and only spend time in service. Since  $E[S] < \infty$  by assumption, jobs spend a finite time (with probability one) at a server.

**s.5.2.9.** Write  $\rho = \lambda/\mu$ . Then, from the formulas for the  $M/M/\infty$  queue, it follows that  $p(n) = e^{-\rho} \rho^n / n!$ . Interestingly, we see that this is equal to  $P(N = n)$  where  $N$  is a Poisson r.v. with parameter  $\rho$ . Thus, the number in the system  $L$  is Poisson distributed with parameter  $\rho$ , thus  $E[L] = \rho$ .

Another way to see that  $E[L] = \rho$  is by noting that in the  $M/M/\infty$  queue jobs do not interact with each other in the queue. When they arrive, there is always a free server available. Since work arrives at rate  $\rho$ , and all jobs are in service simultaneously, the average number of busy servers must also be  $\rho$ .

**s.5.2.10.** Use the figure below. Make sure you understand why  $\mu(2) = 2\mu$  and so on.



From this figure it follows right away that:

$$\begin{aligned}\lambda p(0) &= \mu p(1) \\ \lambda p(1) &= 2\mu p(2) \\ \lambda p(2) &= 2\mu p(3).\end{aligned}$$

Then, from the above, with  $\rho = \lambda/\mu$ :

$$\begin{aligned}p(1) &= \rho p(0), \\ p(2) &= (\rho/2)p(1) = (\rho^2/2)p(0), \\ p(3) &= (\rho/2)p(2) = (\rho^3/4)p(0).\end{aligned}$$

Now we normalize to find  $p(0)$ . Thus, we want that:

$$1 = p(0) + p(1) + p(2) + p(3) = p(0) \left( 1 + \rho + \frac{\rho^2}{2} + \frac{\rho^3}{4} \right),$$

hence,

$$p(0) = (1 + \rho + \rho^2/2 + \rho^3/4)^{-1}.$$



**s.5.2.11.**  $\lambda(n) \equiv \lambda$  for all  $n < c + K$ . When  $n = c + K$ ,  $\lambda(n) = 0$ , since then the system is full, and all arriving jobs will be dropped; in other words, there will still be jobs arriving to the system when  $L = c + K$ , but these jobs will be rejected, hence cannot generate a transition from state  $c + K$  to  $c + K + 1$ . When  $n < c$ ,  $\mu(n) = n\mu$  since only  $n$  servers are active/occupied when the system contains  $n$  jobs. When  $n \geq c$ ,  $\mu(n) = c\mu$ . Thus, using  $\rho = \lambda/(c\mu)$ , for  $n < c$ ,

$$p(n) = \frac{\lambda}{n\mu} p(n-1) = \frac{(\lambda/\mu)^n}{n!} p(0) = \frac{(c\rho)^n}{n!} p(0).$$

For  $c \leq n \leq c + K$  and using the above to get  $p(c-1)$ :

$$\begin{aligned} p(n) &= \frac{\lambda}{c\mu} p(n-1) = \rho p(n-1) = \rho^2 p(n-2) = \dots \\ &= \rho^{n-c+1} p(c-1) = \rho^{n-c+1} \frac{(c\rho)^{c-1}}{(c-1)!} p(0) \\ &= \rho^n \frac{(c)^{c-1}}{(c-1)!} p(0) = \rho^n \frac{(c)^{c-1} c}{(c-1)! c} p(0) = \frac{c^c \rho^n}{c!} p(0). \end{aligned}$$

The normalization is trivial, numerically at least.

**s.5.2.12.** Take  $\lambda(n) = (N-n)\lambda$  and  $\mu(n) = \mu$ , and solve (4.2.7) and (4.2.9). Thus:

$$\begin{aligned} p(n+1) &= \frac{(N-n)\lambda}{\mu} p(n) = \rho(N-n)p(n) \\ &= \rho^2(N-n)(N-(n-1))p(n-1) \\ &= \rho^3(N-n)(N-(n-1))(N-(n-2))p(n-2) \\ &= \rho^{n+1}(N-n)(N-(n-1))\dots(N-(0))p(0) \\ &= \rho^{n+1} \frac{N!}{(N-(n+1))!} p(0). \end{aligned}$$

Next, we need to normalize this. Observe that  $p(N+1) = p(N+2) = \dots = 0$  since there are just  $N$  customers, so that the system can never contain more than  $N$  customers. Thus, we want  $p(0)$  to be such that

$$1 = \sum_{n=0}^N p(n) = p(0) \sum_{n=0}^N \rho^n \frac{N!}{(N-n)!}.$$

We see from this that  $p(0)$  times some constant must be 1. Hence, dividing by this constant, we get

$$p(0) = \left( \sum_{n=0}^N \rho^n \frac{N!}{(N-n)!} \right)^{-1}.$$

I asked WolframAlpha to simplify this, but the answer I got was not particularly revealing.

**s.5.2.13.** A finite calling population occurs for instance at a factory with a number of machines. When a machine breaks down, it becomes a (repair) job at the repair department. Thus, a break down forms an arrival at the repair shop. The mechanics at the repair department form a set of parallel servers. Typically, the number of machines is quite small, 10 or so, and when a machine is ‘down’, i.e., broken, it cannot break again. Hence, when 2, say, machines are in repair, the number of ‘customers’ that can arrive to the queueing system is only 8.

**s.5.2.14.** Take  $\lambda(n) = (N - n)\lambda$  and  $\mu(n) = n\mu$ . Then

$$\begin{aligned} p(n+1) &= \frac{\lambda(n)}{\mu(n+1)} p(n) = \frac{(N-n)\lambda}{(n+1)\mu} p(n) = \frac{(N-n)(N-(n-1))}{(n+1)n} \frac{\lambda^2}{\mu^2} p(n-1) \\ &= \frac{N!}{(N-(n+1))!} \frac{1}{(n+1)!} \rho^{n+1} p(0) = \binom{N}{n+1} \rho^{n+1} p(0). \end{aligned}$$

Hence, after normalization, i.e., requiring that  $\sum_{n=0}^N p(n) = 1$ , so that  $p(0) = \left( \sum_{k=0}^N \rho^k \binom{N}{k} \right)^{-1}$ , the final result becomes

$$p(n) = \frac{\rho^n \binom{N}{n}}{\sum_{k=0}^N \rho^k \binom{N}{k}}.$$

**s.5.2.15.** In a queueing system with balking, customers may decide to balk at a level  $b$ . Thus, whether only  $b$  customers are admitted to the system (i.e., blocked), or balk at level  $b$ , the effect is the same: the number of people in the system remains at or below  $b$ . However, a fraction of the customers may already balk at lower levels, like in the example above, so that the arrival stream is ‘thinned’ due to balking customers. In that respect, a queueing system with balking behaves differently.

**s.5.2.16.** First find  $E[L]$

```
>>> P = [0.4, 0.3, 0.2, 0.05, 0.05]
>>> EL = sum(n * P[n] for n in range(len(P)))
>>> EL
1.05
```

There can only be a queue when a job is in service. Since there is  $m = 1$  server, we subtract  $m$  from the amount of jobs in the system. Before we do this, we need to ensure that  $n - m$  does not become negative. Thus,  $E[L_Q] = \sum_n \max\{n - m, 0\} p(n)$ .

```
>>> m = 1
>>> ELq = sum(max(n - m, 0) * P[n] for n in range(len(P)))
>>> ELq
0.45000000000000007

>>> labda = 10.0 / 60
>>> Wq = ELq / labda # in minutes
>>> Wq
2.7000000000000006
>>> Wq / 60 # in hours
0.04500000000000001

>>> W = EL / labda # in minutes
>>> W
6.300000000000001
>>> W / 60 # in hours
0.10500000000000001
```

Let's use the standard definition of the variance, i.e.,  $V[X] = \sum_i (x_i - E[X])^2 P(X = x_i)$ , for once.

```
>>> from math import sqrt

>>> var_L = sum((n - EL) ** 2 * P[n] for n in range(len(P)))
>>> var_L
1.2475
>>> sqrt(var_L)
1.116915395184434

>>> var_Lq = sum((max(n - m, 0) - ELq) ** 2 * P[n] for n in range(len(P)))
>>> var_Lq
0.6475
>>> sqrt(var_Lq)
0.8046738469715541
```

Service time is sojourn time minus time in queue.

```
>>> ES = W - Wq

>>> rho = labda * ES
>>> rho
0.6
```

The expected number of jobs at the server is the expected number in the system minus the the expected number in queue. Thus,

```
>>> rho = EL - ELq
>>> rho
0.6
```

This checks with the previous line.

The utilization must also be equal to the fraction of time the server is busy.

```
>>> u = 1 - P[0]
>>> u
0.6
```

Yet another way: Suppose we have  $m$  servers. If the system is empty, all  $m$  servers are idle. If the system contains one customer, 1 server is busy, and so on. We therefore define the *utilization of the station* as

$$u = \sum_{n=0}^{\infty} \min\{n, m\} p_n.$$

```
>>> u = sum(min(n, m) * P[n] for n in range(len(P)))
>>> u
0.6000000000000001
```

**s.5.2.17.** Note that state 4 cannot be accessed, because  $\lambda(n) = 0$  for  $n \geq 3$ , in particular  $\lambda(3) = 0$ , so there are no arrivals when the system contains  $n = 3$  jobs.

```

>>> labda = [5, 3, 2, 0, 0]
>>> mu = [0, 2, 3, 4, 4]

>>> p = [1, 0, 0, 0]

>>> p[1] = labda[0] * p[0] / mu[1]
>>> p[2] = labda[1] * p[1] / mu[2]
>>> p[3] = labda[2] * p[2] / mu[3]

>>> print(p)
[1, 2.5, 2.5, 1.25]

>>> norm = sum(p)
>>> p = [x / norm for x in p]
>>> print(p)
[0.13793103448275862, 0.3448275862068966, 0.3448275862068966, 0.1724137931034483]

```

$p(n) \neq (1-\rho)\rho^n$  because the arrival rate is not the same in all states, neither is the service rate.

**s.5.2.18.** First compute  $E[L_Q]$  for the case with a buffer for 2 customers.

```

>>> labda = 6.0
>>> mu = 5.0
>>> rho = labda / mu
>>> c = 1
>>> b = 2

```

Set  $p(n) = \rho^n$  initially, and normalize later. Use the expressions for the  $M(n)/M(n)/1$  queue. Observe that  $\rho > 1$ . Since the size of the system is  $c + b + 1$  is finite, all formulas work for this case too.

There are 4 states in total: 0,1,2,3. (The reason to import numpy here and convert the lists to arrays is to fix the output precision to 3, otherwise we get long floats in the output.)

```

>>> import numpy as np

>>> np.set_printoptions(precision=3)

>>> P = np.array([rho ** n for n in range(c + b + 1)])
>>> P
array([1.    , 1.2   , 1.44 , 1.728])

>>> G = sum(P)
>>> G
5.368

>>> P /= G # normalize
>>> P
array([0.186, 0.224, 0.268, 0.322])

```

```
>>> L = sum(n * P[n] for n in range(len(P)))
>>> L
1.725782414307004

>>> Lq = sum((n - c) * P[n] for n in range(c, len(P)))
>>> Lq
0.9120715350223545
```

The number of jobs served per hour must be equal to the number of jobs accepted, i.e., not lost. The fraction of customers lost is equal to the fraction of customers that sees a full system.

```
>>> lost = labda * P[-1] # the last element of P
>>> lost
1.9314456035767507

>>> accepted = labda * (1.0 - P[-1]) # rate at which jobs are accepted
>>> accepted
4.06855439642325
```

Now increase the buffer  $b$  to 5.

```
>>> b = 5
>>> P = np.array([rho ** n for n in range(c + b + 1)])
>>> P
array([1.    , 1.2   , 1.44  , 1.728, 2.074, 2.488, 2.986])
>>> G = sum(P)
>>> G
12.915903999999998

>>> P /= G # normalize
>>> P
array([0.077, 0.093, 0.111, 0.134, 0.161, 0.193, 0.231])

>>> L = sum(n * P[n] for n in range(len(P)))
>>> L
3.7098374221424995

>>> accepted = labda * (1.0 - P[-1])
>>> accepted
4.6128803682653565
```

**s.5.2.19.** Determine  $E[L]$  and  $E[L_Q]$ .

```
>>> P = [0.4, 0.3, 0.2, 0.05, 0.05]

>>> L = sum(n * P[n] for n in range(len(P)))
>>> L
```

1.05

```
>>> c = 2
>>> Lq = sum((n - c) * P[n] for n in range(c, len(P)))
>>> Lq
0.15000000000000002
```

Using Little's formula, determine  $E[W]$  and  $E[W_Q]$ .

```
>>> labda = 10.0 / 60
>>> Wq = Lq / labda # in minutes
>>> Wq
0.9000000000000001
>>> Wq / 60 # in hours
0.015000000000000003
```

```
>>> W = L / labda
>>> W
6.3000000000000001
```

Determine  $V[L]$  and  $V[L_Q]$ .

```
>>> from math import sqrt

>>> var_L = sum((n - L) ** 2 * P[n] for n in range(len(P)))
>>> var_L
1.2475
>>> sqrt(var_L)
1.116915395184434

>>> var_Lq = sum((max(n - c, 0) - Lq) ** 2 * P[n] for n in range(len(P)))
>>> var_Lq
0.22750000000000004
```

Determine the service time and the utilization.

```
>>> ES = W - Wq
>>> ES # in minutes
5.4
```

The load per server is given by:

```
>>> rho = labda * ES / c
>>> rho
0.45
```

The expected number of busy servers is

```
>>> EBusy = L - Lq
>>> EBusy
0.9
```

For the utilization, see [5.2.16]. We now have  $c = 2x$  servers.

```
>>> u = sum(min(n, c) * P[n] for n in range(len(P)))
>>> u
0.8999999999999999
```

And this matches with the above.

**s.5.2.20.** First the service rates.

```
>>> import numpy as np
>>> from math import factorial

>>> labda = [10.0, 15.0, 15.0, 10.0, 5.0]
>>> c = 2
>>> mn = 2 * np.ones(len(labda) + 1, dtype=int) # number of active servers
>>> mn[0] = 0 # no service if system is empty
>>> mn[1] = 1 # one busy server if just one job present
>>> mu = 5 * mn # service rate is 5 times no of active servers
>>> mu
array([ 0,  5, 10, 10, 10, 10])
```

Since there can be arrivals in states  $0, \dots, 4$ , the system can contain 0 to 5 customers, i.e.,  $p(0), \dots, p(5)$ .

Use the level-crossing result for the  $M(n)/M(n)/1$  queue:

```
>>> P = [1] * (len(labda) + 1)
>>> for i in range(1, len(P)):
...     P[i] = labda[i - 1] / mu[i] * P[i - 1]
...
>>> P = np.array(P) # unnormalized probabilities
>>> P
array([1. , 2. , 3. , 4.5 , 4.5 , 2.25])

>>> G = sum(P) # normalization constant
>>> G
17.25
>>> P /= G # normalize
>>> P
array([0.058, 0.116, 0.174, 0.261, 0.261, 0.13 ])
```

$$\lambda = \sum_n \lambda(n)p(n).$$

```
>>> labdaBar = sum(labda[n] * P[n] for n in range(len(labda)))
>>> labdaBar
8.840579710144928
```

The average number in the system is:

```
>>> Ls = sum(n * P[n] for n in range(len(P)))
>>> Ls
2.942028985507246
```

The average number in queue:

```
>>> c = 2
>>> Lq = sum((n - c) * P[n] for n in range(c, len(P)))
>>> Lq
1.1739130434782608
```

And now the waiting times:

```
>>> Ws = Ls / labdaBar
>>> Ws # time in the system
0.3327868852459016

>>> Wq = Lq / labdaBar
>>> Wq # time in queue
0.13278688524590163
```

**s.5.2.21.** Would one server/person do?

```
>>> labda = 28.0 / 60 # arrivals per minute
>>> ES = 4.0
>>> labda * ES
1.8666666666666667
```

If  $c = 1$ , the load  $\rho = \lambda E[S]/c > 1$  is clearly undesirable for one server. We need at least two servers.

It is not relevant to focus on the time in the system, as time in service needs to be spent anyway. Hence, we focus on the waiting time in queue.

I just convert the formulas of [5.2.3] to Python code. This saves me time during the computations.

```
>>> from math import factorial

>>> def WQ(c, labda, ES):
...     rho = labda * ES / c
...     G = sum([(c * rho) ** n / factorial(n) for n in range(c)])
...     G += (c * rho) ** c / (1.0 - rho) / factorial(c)
...     Lq = (c * rho) ** c / (factorial(c) * G) * rho / (1.0 - rho) ** 2
...     return Lq / labda # Wq, Little's law
...

>>> #
```

Considering the scenario with one server is superfluous as  $\rho > 1$  in that case.  
What is the waiting time for  $c = 2$  servers?



```
>>> WQ(2, 28.0 / 60, 4) # in minutes
27.034482758620694
>>> WQ(2, 28.0 / 60, 4) / 60.0 # in hours
0.4505747126436782
```

What is the waiting time for  $c = 3$  servers?

```
>>> WQ(3, 28.0 / 60, 4) # in minutes
1.3542675591474136
>>> WQ(3, 28.0 / 60, 4) / 60.0 # in hours
0.022571125985790228
```

What is the waiting time for  $c = 4$  servers?

```
>>> WQ(4, 28.0 / 60, 4) # in minutes
0.26778942672317635
>>> WQ(4, 28.0 / 60, 4) / 60.0 # in hours
0.0044631571120529396
```

In the next part of the question we will interpret these numbers.

Since both types of workers cost the same amount of money per unit time, it is best to divide the amount of waiting/idleness equally over both types of workers. I am inclined to reason as follows. The average amount of waiting time done by the maintenance workers per hour is  $\lambda E[W_Q]$ . To see this, note that maintenance workers arrive at rate  $\lambda$ , and each worker waits on average  $E[W_Q]$  minutes. Thus, worker time is wasted at rate  $\lambda E[W_Q]$ . Interestingly, with Little's law,  $E[L_Q] = \lambda E[W_Q]$ , i.e., the rate at which workers waste capacity (i.e. waiting in queue) is  $E[L_Q]$ . On the other hand, the rate of work capacity wasted by the tool crib employees being idle is  $c - \lambda E[S]$ , as  $\lambda E[S]$  is the average number of servers busy, while  $c$  crib servers are available.

As both types of employees are equally expensive, we need to choose  $c$  such that the number of maintenance workers waiting (i.e., being idle because they are waiting in queue), is equal to the number of crib workers being idle. In other words, we search for a  $c$  such that  $E[L_Q] \approx c - \lambda E[S]$  (where, of course,  $E[L_Q]$  depends on  $c$ ).

```
>>> labda = 28.0 / 60
>>> ES = 4.0
>>> c = 2
>>> ELQ = labda * WQ(c, labda, ES)
>>> ELQ
12.616091954022991
>>> c - labda * ES
0.13333333333333333
```

Now the maintenance employees wait more than the tool crib employees.

```
>>> c = 3
>>> ELQ = labda * WQ(c, labda, ES)
>>> ELQ
```

```

0.6319915276021264
>>> c - labda * ES
1.1333333333333333

>>> c = 4
>>> ELQ = labda * WQ(c, labda, ES)
>>> ELQ
0.1249683991374823
>>> c - labda * ES
2.1333333333333333

```

Clearly,  $c = 3$  should do.

**s.5.2.22.** Let  $p_{ij}$  be the fraction of time that the system contains  $i$  riders and  $j$  taxi cabs.

I assume that all members of a party of riders can be served by a single cab (that is, the parties do not exceed the capacity of a cab and all members of a party have the same destination).

For clarity, write  $\mu$  for the rate at which cabs arrive, and  $\lambda$  for the arrival rate of parties of riders.

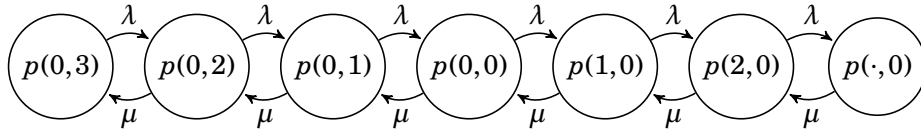
Then the transitions are as in the figure below.

Suppose first that there are 3 taxi cabs.

When a group arrives (at rate  $\lambda$ ), there is one taxi less, and so on, until there are no more taxis left.

Finally, if yet more groups arrive, they have to wait.

When a new taxi arrives, the number of groups is reduced by one, and so on, until there are 3 taxis waiting and no groups of people.



From this figure, we see that

$$\begin{aligned}
 \lambda p_{0,3} &= \mu p_{0,2} \\
 (\lambda + \mu) p_{0,2} &= \mu p_{0,1} + \lambda p_{0,3} \\
 (\lambda + \mu) p_{0,1} &= \mu p_{0,0} + \lambda p_{0,2} \\
 (\lambda + \mu) p_{0,0} &= \mu p_{1,0} + \lambda p_{0,1} \\
 (\lambda + \mu) p_{1,0} &= \mu p_{2,0} + \lambda p_{0,0} \\
 (\lambda + \mu) p_{2,0} &= \mu p_{3,0} + \lambda p_{1,0}
 \end{aligned}$$

and so on. Thus, it is left to compute  $p_{ij}$ . Observe from this scheme, or the above figure, that the situation with the taxis correspond to an  $M/M/1$  queue, only the states have a ‘different name’. Let  $q$  be the number of jobs in an  $M/M/1$  queue. Some thought will reveal that the queueing system with cabs and parties can be mapped to an equivalent  $M/M/1$  queueing system. In fact, consider the following table

$j$	$i$	$q$
3	0	0
2	0	1
1	0	2
0	0	3
0	1	4
0	2	5

and so on. Therefore, in general, it must be that

$$q = 3 - j + i.$$

From the M/M/1 queue we know right away that  $p_q = \rho^q(1 - \rho)$ . With the above relation we can therefore immediately find that  $p_{ij} = \rho^{3-j+i}(1 - \rho)$ , save that  $i$  and  $j$  must satisfy the constraints imposed by the model.

Second, the expected number of cabs waiting must be

$$1p_{0,1} + 2p_{0,2} + 3p_{0,3}$$

and the expected number of parties waiting must be  $\sum_{j=1}^{\infty} jp_{j,0}$ .

```
>>> labda = 12.0 # per hour
>>> mu = 15.0 # per hour
>>> rho = labda / mu

>>> def p(i, j):
...     q = 3 - j + i
...     return rho ** q * (1.0 - rho)
...

>>> #
```

Expected number of cabs waiting:

```
>>> Lc = sum(j * p(0, j) for j in range(0, 4))
>>> # Recall this sums up to 4, not including 4
>>> Lc
1.0479999999999998
```

To compute the expected number of parties waiting we formally have to sum to infinity. Rather than doing the algebra, I chose to truncate the summation at an  $i$  such that  $\rho^i \ll 1$ , i.e., negligible. Truncating at 30 seems reasonable enough:

```
>>> trunc = 30
>>> rho ** trunc
0.0012379400392853823
```

At second thought this is not yet really small.

```
>>> trunc = 50
>>> rho ** trunc
1.4272476927059638e-05
```

This is better. Now go for what we want to know:

```
>>> Lp = sum(i * p(i, 0) for i in range(trunc))
>>> Lp
2.0476053945579213
```

For the last part: This is tricky. I first, naively, computed  $W_q = L_c/\mu$ . This seems to make sense, as cabs arrive at rate  $\mu$ , so that this expression follows from a standard application of Little's law. However, this is wrong, of course. When using Little's law to relate the number of jobs in queue (i.e., in the M/M/1 queue) and the queueing time we need to use  $\lambda$ , not  $\mu$ . Similarly (and more formally by the mapping developed in part a), for our cab system we also need to use  $\lambda$ .

```
>>> Wq = Lc / labda
>>> Wq
0.08733333333333332
```

Thinking in templates is often useful, but makes one sloppy...

What would be the impact of allowing 4 cabs? Funny question, and with the above, trivial to answer.

```
>>> def p(i, j):
...     q = 4 - j + i
...     return rho ** q * (1.0 - rho)
...

>>> #

>>> Lc = sum(j * p(0, j) for j in range(0, 4))
>>> Lc
0.8383999999999999

>>> Lp = sum(i * p(i, 0) for i in range(trunc))
>>> Lp
1.638084315646337
```

**s.5.2.23.** Now we have the standard M/M/1 queue. The number of parties waiting must be  $E[L]$  of the M/M/1 queue. We can use Little's law to compute the waiting time.

**s.5.2.24.** Actually, you don't have to use PASTA. So why is that? For instance,  $\sum_{i=1}^3 i p(0, i)$  is the number of taxis waiting, and this is a time average (since we use  $p(0, i)$ ). In the proof of Little's law, it is also clear that the  $E[L]$  is a time average. Also, in the proof of Little's law, we compute the waiting time as  $E[W] = \lim_{n \rightarrow \infty} n^{-1} \sum_{k=1}^n W_k$ , where  $W_k$  is the waiting time as perceived by the  $k$ th job. Thus, here  $E[W]$  is the average as observed by jobs.

This was an old exam question. Some students used the PK-formula, see Section 5.4, to compute the average waiting time. The derivation of this *does* depend on the PASTA property.

**s.5.2.25.** We concentrate on departure epochs of the taxis. Thus the  $k$ th period is the time between the departure of taxi  $k - 1$  and taxi  $k$ . During the  $k$ th epoch  $a_k$  batches can arrive.

The system starts with  $a_0$  batches in queue.

Suppose that the first batch contains 5 riders. Then the first taxi takes 4 riders, and 1 rider of the batch remains. This 1 rider will take the next taxi, and no riders of other groups can join because the riders of different parties have different destinations. Once all riders of a party are served, the next party in line can move to the ‘server’ and wait to be served.

The recursions are as follows; realize that the order in which you carry out these recursions is important. The meaning of each line is explained below the line.

$$d_s = \min\{Q_s, 4\},$$

the number of riders that can depart from the party of riders in service,

$$Q'_s = Q_s - d_s,$$

the remaining of number riders of a party after being served by one taxi,

$$Q = Q + a_k,$$

the number of parties in queue (not in the system) just after the arrival of the batches during the  $k$ th interdeparture time,

$$pd_q = \min\{Q, \mathbb{1}_{Q'_s=0}\},$$

only move a party from the queue if ‘the server is free’,

$$Q = Q - d_q,$$

move the party from the queue to the server, if allowed,

$$Q_s = Q'_s \mathbb{1}_{Q'_s>0} + B_b \mathbb{1}_{Q'_s=0},$$

if the server is not free, the number of riders is equal to  $Q'_s$ , otherwise send the  $b$ th batch to the server,

$$b = b + \mathbb{1}_{Q'_s=0},$$

if the server is free, move the index of the batch in service to the next batch to be served once the server becomes free again.

In the code  $A_k$  corresponds to a list of batches arriving on the  $k$ th day,  $B_i$  to the size of the  $i$ th batch, and  $a_k$  to the number of batches arriving on the  $k$ th period. I used the `pysnooper` module to debug the code. It is quite hard to get it right.

```
A = [[5, 3, 4], [3], [6], [1, 1], [], [2], [], [], [3], []]
```

```
a = [len(A[i]) for i in range(len(A))]
```

```
B = [item for sublist in A for item in sublist]
```

```

Qs = 0
b = 0
Q = 0

for k in range(len(a)):
    ds = min(Qs, 4)
    Qs_p = Qs - ds
    Q += a[k]
    dq = min(Q, 1 * (Qs_p == 0))
    Q -= dq
    Qs = (Qs_p > 0) * Qs_p + (Qs_p == 0) * B[b]
    b += Qs_p == 0
    print(f"ds={ds}, Qs_p={Qs_p}, dq={dq}, Q={Q}, Qs={Qs}, b={b}")

#

```

### 5.3 $M^X/M/1$ QUEUE: EXPECTED WAITING TIME

Sometimes jobs arrive in batches, rather than as single units. For instance, when a car or a bus arrives at a fast-food restaurant, a batch consists of the number of people in the vehicle. When the batches arrive as a Poisson process and the individual items within a batch have exponential service times we denote such queueing systems by the shorthand  $M^X/M/1$ . We derive expressions for the load and the expected waiting time and queue length for this queueing model.

#### *Theory and Exercises*

Assume that jobs arrive as a Poisson process with rate  $\lambda$  and each *job* contains multiple *items*. Let  $A_k$  be the arrival time of job  $k$  and  $A(t)$  the number of (job) arrivals up to time  $t$ . Denote by  $B_k$  the batch size of the  $k$ th job, i.e., the number of items of job  $k$ . We assume that  $\{B_k\}$  is a sequence of independent discrete random variables each distributed as the generic random variable  $B$ . Let  $f(k) = P(B = k)$  be given; let the survivor function be  $G(k) = P(B > k)$ . The service time of each item is  $E[S]$ .

**5.3.1.** *Explain that the average time to serve an entire batch is  $E[B] E[S]$ , so that the load must given by  $\rho = \lambda E[B] E[S]$ .*

The aim of the remainder of the section is to derive a cornerstone of queueing theory, which is the following formula for the expected time an item spends in queue:

$$E[W_Q] = \frac{1 + C_s^2}{2} \frac{\rho}{1 - \rho} E[B] E[S] + \frac{1}{2} \frac{\rho}{1 - \rho} E[S], \quad (5.3.1)$$

where  $C_s^2$  is the SCV of the batch size distribution. By applying (4.4.3), it follows right away that the expected number of items in the system takes the form

$$E[L] = \frac{E[W_Q]}{E[S]} = \frac{1 + C_s^2}{2} \frac{\rho}{1 - \rho} E[B] + \frac{1}{2} \frac{\rho}{1 - \rho}. \quad (5.3.2)$$

Note that  $\rho < 1$  is required, as usual.

Before deriving the above, let us try to use it.

**5.3.2.** What is  $E[L]$  in case  $B_k = 3$  always, and  $\lambda = 1$ ,  $\mu = 6$ ?

**5.3.3.** If the batch size is geometrically distributed with success probability  $p$ , what is  $E[L]$ ?

**5.3.4.** A common operational problem is a machine that receives batches of various sizes. Management likes to know how a reduction of the variability of the batch sizes would affect the average queueing time. Suppose, for the sake of an example, that the batch size

$$P(B = 1) = P(B = 2) = P(B = 3) = \frac{1}{3}.$$

Batches arrive at rate 1 per hour. The average processing time for an item is 25 minutes. Compute by how much the number of items in the system would decrease if batch sizes were constant and equal to 2; hence the load is the same in both cases.

**5.3.5.** Show that when the batch size is 1, the expression  $E[L(M^X/M/1)]$ , i.e., the system length for the  $M^X/M/1$  queue, reduces to  $E[L(M/M/1)]$ , i.e., the system length for the  $M/M/1$  queue. Realize the importance of such checks.

**5.3.6.** Show that  $E[W_Q(M^X/M/1)] \geq E[W_Q(M/M/1)]$  when the loads are the same. What do you conclude? (This solution of this exercise is more useful than you might think.)

Let us now focus on deriving (5.3.1). Assume that an arriving batch joins the end of the queue (if present), and once the queue in front of it has been cleared, it moves in its entirety to the server. Thus, all items in one batch spend the same time in queue. Once the batch moves to the server, the server processes the items one after another until the batch is empty. Write  $E[L_Q^B]$  for the number of batches in queue and  $E[L_S^B]$  for the number of items of the job (if any) at the server. Observe first that the average time an item spends in queue is

$$E[W_Q] = E[L] E[S] = \left( E[L_Q^B] E[B] + E[L_S^B] \right) E[S].$$

We also see that the average time a batch spends in queue is

$$E[W_Q^B] = E[L_Q^B] E[B] E[S] + E[L_S^B] E[S].$$

Hence,  $E[W_Q] = E[W_Q^B]$ .

**5.3.7.** Use Little's law to show that

$$E[W_Q^B] = \frac{E[L_S^B]}{1 - \rho} E[S], \text{ hence } E[L] = \frac{E[L_S^B]}{1 - \rho}. \quad (5.3.3)$$

Clearly, we are done if we can find an expression for  $E[L_S^B]$ . For this we can use the renewal reward theorem; in fact, we can use [4.1.1] as inspiration. (Solve this exercise if you have not done yet.) Define  $Y(t) = \int_0^t L_S^B(s) ds$  to see that  $E[L_S^B] = Y = \lim_{t \rightarrow \infty} Y(t)/t$ .

**5.3.8.** Let  $X_k = Y(D_k) - Y(D_{k-1})$ . Then  $E[X] = \lim_{n \rightarrow \infty} n^{-1} \sum_{k=1}^n X_k$ . With this, show that

$$E[X] = \frac{E[B^2] + E[B]}{2} E[S].$$

**5.3.9.** Use the renewal reward theorem with sampling epochs  $T_k = D_k$  to prove that

$$\mathbb{E}[L_S^B] = \lambda \frac{\mathbb{E}[B^2]}{2} \mathbb{E}[S] + \frac{\rho}{2}.$$

**5.3.10.** Show that

$$\frac{\mathbb{E}[B^2]}{(\mathbb{E}[B])^2} = 1 + C_s^2$$

**5.3.11.** Use the above to derive (5.3.1).

*Hints*

**h.5.3.2.** Use (5.3.2). What are  $\mathbb{E}[B^2]$ ,  $\mathbb{E}[B]$  and  $\mathbb{V}[B]$  for this case?

**h.5.3.3.**  $f_k = q^{k-1}p$  with  $q = 1 - p$ . Use generating functions to compute  $\mathbb{E}[B]$  and  $\mathbb{E}[B^2]$ .

**h.5.3.5.** What is the distribution of the batch size  $B$  for the  $M/M/1$  queue?

**h.5.3.6.** Use [5.3.5] and Jensen's inequality.

**h.5.3.7.** Use Little's law in  $\mathbb{E}[W_Q^B] = \mathbb{E}[L_Q^B] \mathbb{E}[B] \mathbb{E}[S] + \mathbb{E}[L_S^B] \mathbb{E}[S]$ .

**h.5.3.8.** Realize that  $X_k = \int_{D_{k-1}}^{D_k} L_S^B(s) ds$  is precisely the amount added to  $Y$  by the  $k$ th batch. Suppose now that the size of the  $k$ th batch is  $B_k = 2$ . Writing  $S_1$  for the service time of the first item of this batch and  $S_2$  for the service of the second item, we see that  $X_k = 2 \cdot S_1 + 1 \cdot S_2$ . Now consider general batch sizes  $B$ ; realize that we sum over a random number of random variables (because  $B$  is a random variable).

**h.5.3.9.** Solve [5.3.8] first.

**h.5.3.11.** Use [5.3.9] and [5.3.10]

*Solutions*

**s.5.3.1.** The expected service time of a batch is  $\mathbb{E}[B] \mathbb{E}[S]$ , because  $\mathbb{E}[B]$  is the expected number of items in a job, and the expected service time of each item is  $\mathbb{E}[S]$ . Since jobs arrive at rate  $\lambda$ , work arrives at rate  $\lambda \mathbb{E}[B] \mathbb{E}[S]$ .

**s.5.3.2.** As  $B$  is constant and equal to 3,  $\mathbb{E}[B^2] = 9$ . Hence,  $\mathbb{V}[B] = 0$ , which implies  $C_s^2 = 0$ . Also,  $\rho = \lambda \mathbb{E}[B] / \mu = 1 \cdot 3 / 6 = 1/2$ . Hence,

$$\mathbb{E}[L] = \frac{1}{2} \frac{1/2}{1 - 1/2} \cdot 3 + \frac{1}{2} \frac{1/2}{1 - 1/2}.$$

**s.5.3.3.** We need  $\mathbb{V}[B]$  and  $\mathbb{E}[B]$ . Consider

$$\begin{aligned} M_B(s) &= \mathbb{E}[e^{sB}] = \sum_{k=0}^{\infty} e^{sk} \mathbb{P}(B = k) \\ &= \sum_{k=0}^{\infty} e^{sk} p q^{k-1} = \frac{p}{q} \sum_{k=0}^{\infty} (q e^s)^k = \frac{p}{q} \frac{1}{1 - q e^s}, \end{aligned}$$



$$\begin{aligned}
E[B] &= M'_B(0) = \frac{p}{q} \frac{q}{(1-qe^s)^2} \Big|_{s=0} = \frac{p}{(1-q)^2} = \frac{1}{p}, \\
E[B^2] &= M''_B(0) = \frac{2}{p^2} - \frac{1}{p}, \\
V[B] &= E[B^2] - (E[B])^2 = \frac{2}{p^2} - \frac{1}{p} - \frac{1}{p^2} = \frac{1}{p^2} - \frac{1}{p}, \\
C_s^2 &= \frac{V[B]}{(E[B])^2} = p^2 \left( \frac{1}{p^2} - \frac{1}{p} \right) = 1 - p, \\
(1 + C_s^2)/2 &= 1 - p/2, \\
E[L] &= \left(1 - \frac{p}{2}\right) \frac{\rho}{1-\rho} \frac{1}{p} + \frac{1}{2} \frac{\rho}{1-\rho} = \frac{\rho}{1-\rho} \frac{1}{p}.
\end{aligned}$$

Can we check this in a simple way? If  $P(B=1) = f_1 = p = 1$ , then  $E[L] = \rho/(1-\rho)$ . Thus, we get the result for the  $M/M/1$  queue. The result is at least consistent with earlier work.

**s.5.3.4.** Start with the simple case,  $B \equiv 2$ . Then  $V[B] = 0$  and  $E[B] = 2$ . The load is  $\rho = \lambda E[B] E[S] = 1 \cdot 2 \cdot 25/60 = 5/6$ . Hence,

$$E[L] = \frac{1}{2} \frac{5/6}{1/6} 2 + \frac{1}{2} \frac{5/6}{1/6} = 5 + \frac{5}{2}.$$

Now the other case.  $E[B^2] = (1 + 4 + 9)/3 = 14/3$ . Hence,  $V[B] = 14/3 - 4 = 2/3$ . Hence,

$$C_s^2 = \frac{V[B]}{(E[B])^2} = \frac{2/3}{4} = \frac{1}{6}.$$

And thus,

$$E[L] = \frac{1 + 1/6}{2} \frac{5/6}{1/6} 2 + \frac{1}{2} \frac{5/6}{1/6} = \frac{7}{6} 5 + \frac{5}{2}.$$

If we divide these two answers, we see that the ratio between  $E[L]$  for both answers is 10/9. In other words, we can reduce about 10% of the number of items in the system by working in fixed batch sizes.

Observe how easy it is with these models to get insight into the order of magnitude of queue length reductions or waiting times that can be achieved with changing work habits, such as making batch sizes constant rather than allowing them to vary. Observe also that it is up to management to decide whether such reductions outweigh any efforts to reduce the variation in batch sizes.

**s.5.3.5.** For the  $M/M/1$  queue, each job contains just one item. Thus,  $B \equiv 1$ , hence  $P(B=1) = 1$ ,  $E[B^2] = E[B] = 1$ . Therefore,  $E[L_S^B(M/M/1)] = \rho$ , and  $E[L(M/M/1)] = \rho/(1-\rho)$ .

**s.5.3.6.**

$$\frac{E[W_Q(M^X/M/1)]}{E[W_Q(M/M/1)]} = \frac{E[L_S(M^X/M/1)]}{E[L_S(M/M/1)]} = \frac{E[L_S(M^X/M/1)]}{\rho} = \frac{E[B^2]}{2E[B]} + \frac{1}{2}.$$

With this we can check whether this condition

$$1 \leq \frac{E[W_Q(M^X/M/1)]}{E[W_Q(M/M/1)]} = \frac{E[B^2]}{2E[B]} + \frac{1}{2}$$

is always true. Clearly, it reduces to

$$E[B] \leq E[B^2].$$

Multiply this by  $E[B]$  for reasons to become clear presently to get

$$(E[B])^2 \leq E[B^2] E[B].$$

So, the initial inequality is converted to this, and we like to know whether this always true.

To see this, we can use Jensen's inequality  $\phi(E[X]) \leq E[\phi(X)]$  when  $\phi$  is convex. In this case take  $\phi(x) = x^2$ , so that Jensen's inequality states that  $(E[B])^2 \leq E[B^2]$ . (BTW, note that Jensen's inequality implies that  $V[X] = E[X^2] - (E[X])^2 \geq 0$ .) Now noting that  $B \geq 1$ , as a job minimally contains one item, we get

$$\begin{aligned} (E[B])^2 &\leq E[B^2], \quad \text{by Jensen's inequality} \\ &\leq E[B^2] E[B], \quad \text{as } B \geq 1. \end{aligned}$$

Clearly, this is the inequality we tried to show. As a result,

$$1 \leq \frac{E[W_Q(M^X/M/1)]}{E[W_Q(M/M/1)]}$$

for all  $B$ .

In conclusion, if work arrives in batches, the average number of jobs in the system increases, hence the average waiting time increases.

### s.5.3.7.

$$\begin{aligned} E[W_Q^B] &= E[L_Q^B] E[B] E[S] + E[L_S^B] E[S] \\ &= \lambda E[W_Q^B] E[B] E[S] + E[L_S^B] E[S] \\ &= E[W_Q^B] \rho + E[L_S^B] E[S]. \end{aligned}$$

Now simplify. For the second relation, use that  $E[W_Q^B] = E[W_Q] = E[L] E[S]$ .

**s.5.3.8.** Check the hint. Now, suppose that the  $k$ th batch size is  $B$  (a random variable), then

$$\begin{aligned} X_k &= \int_{D_{k-1}}^{D_k} L_S^B(s) ds \\ &= BS_1 + (B-1)S_2 + \cdots + 1S_B. \end{aligned}$$

Realize that the batch sizes are i.i.d., as are the service times, and the batch sizes and service times are also independent (by assumption). This implies that all  $X_k$  are distributed as a common random variable  $X = BS_1 + (B-1)S_2 + \cdots + 1S_B$ . With this,

$$\begin{aligned} E[X] &= E[BS_1 + (B-1)S_2 + \cdots + 1S_B] \\ &= E\left[\sum_{j=1}^B jS_{B+1-j}\right] \\ &= E\left[\sum_{j=1}^B j\right] E[S], \quad \text{by indep. of } S \text{ and } B \end{aligned}$$

$$\begin{aligned}
&= E[B(B+1)/2] E[S] \\
&= \frac{E[B^2] + E[B]}{2} E[S].
\end{aligned}$$

Here you should be careful: observe that we sum over a *random* number of summands. In such cases, it is not allowed to use the rule that the expectation of a sum is a sum of expectations!

**s.5.3.9.** From the above and [5.3.8],

$$E[L_S^B] = Y = \delta X = \lambda \frac{E[B^2] + E[B]}{2} E[S] = \lambda \frac{E[B^2]}{2} E[S] + \frac{\rho}{2},$$

where we use that  $\rho = \lambda E[B] E[S]$  and  $\delta = \lambda$  (rate-stability).

**s.5.3.10.** We have

$$\begin{aligned}
\frac{E[B^2]}{(E[B])^2} &= \frac{E[B^2] - (E[B])^2 + (E[B])^2}{(E[B])^2} \\
&= \frac{V[B] + (E[B])^2}{(E[B])^2} = C_s^2 + 1.
\end{aligned}$$

**s.5.3.11.** Starting from (5.3.3), and using the intermediate results from the hints, i.e., previous exercises,

$$\begin{aligned}
E[W_Q^B] &= \frac{E[L_S^B]}{1-\rho} E[S] \\
&= \left( \lambda \frac{E[B^2]}{2} E[S] + \frac{\rho}{2} \right) \frac{E[S]}{1-\rho} \\
&= \left( \frac{E[B^2]}{2E[B]} \rho + \frac{\rho}{2} \right) \frac{E[S]}{1-\rho} \\
&= \left( \frac{E[B^2]}{2(E[B])^2} \rho E[B] + \frac{\rho}{2} \right) \frac{E[S]}{1-\rho} \\
&= \frac{1+C_s^2}{2} \frac{\rho}{1-\rho} E[B] E[S] + \frac{1}{2} \frac{\rho}{1-\rho} E[S].
\end{aligned}$$

## 5.4 M/G/1 QUEUE: EXPECTED WAITING TIME

In many practical single-server queueing systems the service times are not really well approximated by the exponential distribution. The *M/G/1* queue then becomes a better model than the *M/M/1* queue. In this section we first present a formula to compute the average waiting time in queue for the *M/G/1* queue, and then we derive it by means of sample path arguments. The derivation is also of general interest as it develops some general results of renewal theory.

### Theory and Exercises

The fundamentally important *Pollaczek-Khinchine formula*, or *PK formula*, for the average waiting time in queue for the *M/G/1* queue has the form

$$E[W_Q] = \frac{1+C_s^2}{2} \frac{\rho}{1-\rho} E[S]. \quad (5.4.1)$$

Before deriving this formula, let us apply it.

**5.4.1.** Show that when services are exponential, the expected waiting time  $E[W_Q(M/G/1)]$  reduces to  $E[W_Q(M/M/1)]$ .

**5.4.2.** Compute  $E[W_Q]$  and  $E[L]$  for the  $M/D/1$  queue.

**5.4.3.** Compute  $E[L]$  for the  $M/G/1$  queue with  $S \sim U[0, \alpha]$ .

**5.4.4.** A queueing system receives Poisson arrivals at the rate of 5 per hour. The single server has a uniform service time distribution, with a range of 4 minutes to 6 minutes. Determine  $E[L_Q]$ ,  $E[L]$ ,  $E[W_Q]$ ,  $E[W]$ .

**5.4.5.** Consider a workstation with just one machine. We model the job arrival process as a Poisson process with rate  $\lambda = 3$  per day. The average service time  $E[S] = 2$  hours,  $C_s^2 = 1/2$ , and the shop is open for 8 hours. What is  $E[W_Q]$ ?

Suppose the expected waiting time has to be reduced to 1h. How to achieve this?

**5.4.6** (Hall 5.16). The manager of a small firm would like to determine which of two people to hire. One employee is fast, on average, but somewhat inconsistent. The other is a bit slower, but very consistent. The first has a mean service time of 2 minutes, with a standard deviation of 1 minute. The second has a mean service time of 2.1 minutes, with a standard deviation of 0.1 minutes. If the arrival rate is Poisson with rate 20 per hour, which employee would minimize  $E[L_Q]$ ? Which would minimize  $E[L]$ ?

**5.4.7.** Show that for the  $M/G/1$  queue, the expected idle time is  $E[I] = 1/\lambda$ .

**5.4.8.** Express the utilization of the  $M/G/1/1$  queue in terms of  $\lambda$  and  $E[S]$ .

**5.4.9.** For the  $M/G/1/1$  queue what is the fraction of jobs rejected?

**5.4.10.** Why is the fraction of lost jobs at a  $M/G/1/1$  queue not necessarily the same as for a  $G/G/1/1$  queue with the same load?

To derive the PK-formula, suppose at first that we know the expected remaining service time  $E[S_r]$ , i.e., the expected time it takes to complete the job in service, if present, at the time a job arrives.

**5.4.11.** Show for the  $M/G/1$  queue that the expected time in queue is

$$E[W_Q] = E[S_r] + E[L_Q] E[S]. \quad (5.4.2)$$

**5.4.12.** Show that, given  $E[S_r]$ ,

$$E[W_Q] = \frac{E[S_r]}{1 - \rho}. \quad (5.4.3)$$

It remains to compute the average remaining service time  $E[S_r]$  for generally distributed service times. Just like in Section 5.3 we use the renewal reward theorem. Consider the  $k$ th job of some sample path of the  $M/G/1$  queueing process. Let its service time start at time  $\tilde{A}_k$  so that it departs at time  $D_k = \tilde{A}_k + S_k$ .

**5.4.13.** Use Fig. 15 to explain that the remaining service time of job  $k$  at time  $s$  is given by  $(D_k - s) \mathbb{1}_{\tilde{A}_k \leq s < D_k}$ . With this, explain that

$$Y(t) = \int_0^t (D_{D(s)+1} - s) \mathbb{1}_{L(s) > 0} ds$$

is the total remaining service time as seen by the server up to  $t$ .

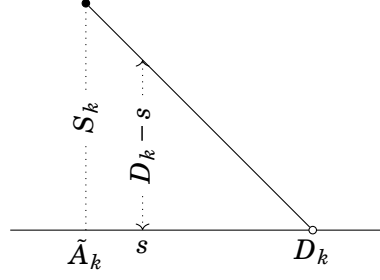


Figure 15: Remaining service time.

**5.4.14.** Apply the renewal reward theorem to the result of [5.4.13] to prove that

$$E[S_r] = \frac{\lambda}{2} E[S^2]. \quad (5.4.4)$$

Then simplify to get (5.4.1).

**5.4.15.** Use  $C_s^2 = V[S]/(E[S])^2$  to show that  $\lambda E[S^2] = (1 + C_s^2)\rho E[S]$ .

**5.4.16.** Use the PASTA property to show that

$$E[S_r] = \rho E[S_r | S_r > 0]. \quad (5.4.5)$$

**5.4.17.** It is an easy mistake to think that  $E[S_r] = E[S]$  when service times are exponential. Why is this wrong?

**5.4.18.** Show from (5.4.4) that

$$E[S_r | S_r > 0] = \frac{E[S^2]}{2E[S]}. \quad (5.4.6)$$

**5.4.19.** When  $V[S] = 0$ , is it true that

$$E[S_r | S_r > 0] = \frac{E[S^2]}{2E[S]} \implies E[S_r | S_r > 0] = \frac{E[S]}{2}?$$

**5.4.20.** What is  $E[S_r | S_r > 0]$  for the M/D/1 queue?

**5.4.21.** Show that  $E[S_r | S_r > 0] = \alpha/3$  when  $S \sim U[0, \alpha]$ .

**5.4.22.** Show that  $E[S_r | S_r > 0] = \mu^{-1}$  when  $S \sim \text{Exp}(\mu)$ .

*Hints*

**h.5.4.2.** Use that  $V[S] = 0$  for the M/D/1 queue.

**h.5.4.3.** Integrate  $\alpha^{-1} \int_0^\alpha x dx$ , and likewise for the second moment.

**h.5.4.7.** What is the average time between two arrivals? Observe that the inter-arrivals are memoryless, hence the average time until the next arrival after the server becomes idle is also  $1/\lambda$ .

**h.5.4.9.** The rate of accepted jobs is  $\lambda\pi(0)$ . What is the load of these jobs? Equate this to  $1 - \pi(0)$  as this must also be the load. Then solve for  $\pi(0)$ .

**h.5.4.10.** Provide an example.

**h.5.4.11.** What happens when you enter a queue? First you have to wait until the job in service (if there is any) completes, and then you have to wait for the queue to clear.

**h.5.4.12.** Use [5.4.11]. Then use Little's law to see that  $E[L_Q] = \lambda E[W_Q]$ . Substitute this in (5.4.2) and simplify.

**h.5.4.14.** Choose  $T_k = D_k$  as epochs in the renewal reward theorem.

**h.5.4.16.** Use [6.3.3].

**h.5.4.17.** Realize again that  $E[S_r]$  includes the jobs that arrive at an empty system.

**h.5.4.19.**  $V[S] = 0$  implies that  $S$  is deterministic.

### Solutions

**s.5.4.1.** Since the SCV for the exponential distribution is 1, hence  $C_s^2 = 1$ , we get

$$E[W_Q] = \frac{1+1}{2} \frac{\rho}{1-\rho} E[S] = \frac{\rho}{1-\rho} E[S],$$

which we also found in a previous section.

**s.5.4.2.** For the  $M/D/1$  the service time is deterministic. Thus, the service time  $S = T$  always. Hence,  $E[S] = T$  and  $E[S^2] = T^2$ , hence  $V[S] = E[S^2] - (E[S])^2 = 0$ , hence  $C_s^2 = 0$ . From the Pollaczek-Khinchine formula, the first term is  $1 + C_s^2 = 1$  for the  $M/D/1$  queue, and  $1 + C_s^2 = 2$  for the  $M/M/1$  queue. Hence,

$$E[W_Q(M/D/1)] = \frac{E[W_Q(M/M/1)]}{2}.$$

Therefore,

$$\begin{aligned} E[L(M/D/1)] &= E[L_Q(M/D/1)] + \rho = \frac{E[L_Q(M/M/1)]}{2} + \rho \\ &= \frac{\rho^2}{2(1-\rho)} + \rho = \frac{\rho(2-\rho)}{2(1-\rho)}. \end{aligned}$$

**s.5.4.3.**

$$\begin{aligned} E[S] &= \alpha/2, \\ E[S^2] &= \int_0^\alpha x^2 dx / \alpha = \alpha^2/3, \\ V[S] &= \alpha^2/3 - \alpha^2/4 = \alpha^2/12, \\ C_s^2 &= (\alpha^2/12)/(\alpha^2/4) = 1/3, \\ \rho &= \lambda\alpha/2, \\ E[W_Q] &= \frac{1+C_s^2}{2} \frac{\lambda\alpha/2}{1-\lambda\alpha/2} \frac{\alpha}{2}, \\ E[W] &= E[W_Q] + \frac{\alpha}{2}, \\ E[L] &= \lambda E[W]. \end{aligned}$$

**s.5.4.4.** First the load.

```

>>> labda = 5./60 # arrivals per minute
>>> a = 4.
>>> b = 6.
>>> ES = (a+b)/2. # service time in minutes
>>> rho = labda*ES
>>> rho
0.41666666666666663

```

Next, the variance and SCV. With this the waiting times follow right away.

```

>>> Var = (b-a)*(b-a)/12.
>>> SCV = Var/(ES**2)

>>> Wq = (1+SCV)/2.*rho/(1.-rho)*ES
>>> Wq # in minutes
1.8095238095238095
>>> Wq/60. # in hour
0.03015873015873016

>>> W = Wq + ES
>>> W
6.809523809523809
>>> Lq = labda*Wq
>>> Lq
0.15079365079365079
>>> L = labda*W
>>> L
0.5674603174603174

```

**s.5.4.5.**  $E[W_Q] = 4.5$  h.  $\rho = \lambda E[S] = (3/8) \cdot 2 = 3/4$ .

One way to increase the capacity/reduce the average service time is to choose  $E[S] = 1$  hour and reduce  $C_s^2$  to  $1/4$ . Of course, there are many more ways. Reducing  $C_s^2$  to zero is (nearly) impossible or very costly. Hence,  $\rho$  must go down, i.e., capacity must go up or service time must go down. Another possibility is to plan the arrival of jobs, i.e., reduce the variability in the arrival process. However, typically this is not possible. For instance, would you accept this as a customer?

**s.5.4.6.** The arrival process is assumed to be Poisson. There is also just one server. Hence, we can use the PK formula to compute the average queue length.

```

>>> labda = 20./60 # per minute
>>> ES = 2. # minutes
>>> sigma = 1.
>>> SCV = sigma*sigma/(ES*ES)
>>> rho = labda*ES

```

```

>>> rho
0.6666666666666666

>>> Wq = (1+SCV)/2 * rho/(1-rho) * ES
>>> Wq
2.4999999999999996
>>> Lq = labda * Wq # Little's law
>>> Lq
0.8333333333333331
>>> W = Wq + ES
>>> W
4.5
>>> L = labda * W
>>> L
1.5

>>> ES = 2.1
>>> SD = 0.1
>>> SCV = SD**2/ES**2
>>> rho = labda*ES
>>> rho
0.7

>>> Lq = rho**2/(1.-rho)*(1.+SCV)/2.
>>> Lq
0.8185185185185182
>>> L = rho + Lq
>>> L
1.5185185185185182

```

**s.5.4.7.** Since the inter-arrival times are memoryless, the expected time to the first arrival after the system becomes empty is also  $E[X] = 1/\lambda$ .

**s.5.4.8.** The system can contain at most 1 job. Necessarily, if the system contains a job, this job must be in service. All jobs that arrive while the server is busy are rejected. Just after a departure, the average time until the next arrival is  $1/\lambda$ , and then a new service starts with an average duration of  $E[S]$ . After this departure, a new cycle starts. Thus, the utilization is  $E[S]/(1/\lambda + E[S]) = \lambda E[S]/(1 + \lambda E[S])$ . Since not all jobs are accepted, the utilization  $\rho$  cannot be equal to  $\lambda E[S]$ .

**s.5.4.9.** Let  $p(0)$  be the fraction of time the server is idle. By PASTA,  $\pi(0) = p(0)$ . Thus, the rate of accepted jobs is  $\lambda\pi(0)$ . Therefore, the departure rate  $\delta = \lambda\pi(0)$ . The loss rate is  $\lambda - \delta = \lambda(1 - \pi(0))$ .

Since  $\lambda\pi(0)$  is the rate at which jobs enter the system, the load must be  $\lambda\pi(0)E[S]$ . Since the load is also  $1 - \pi(0)$ , it follows from equating that

$$\lambda\pi(0)E[S] = 1 - \pi(0) \iff \pi(0) = \frac{1}{1 + \lambda E[S]} \iff 1 - \pi(0) = \frac{\lambda E[S]}{1 + \lambda E[S]},$$



which is the same as in the previous problem.

Note that  $\delta < \lambda$ , as it should be the case.

**s.5.4.10.** Typically, in the  $G/G/1$  queue, the arrivals do not see time-averages. Consequently, the fraction of arrivals that are blocked is not necessarily equal to the utilization  $\rho$ .

Again, take jobs with a duration 59 minutes and inter-arrival times of 1 hour. The load is 59/60, but no job is lost, also not in the  $G/G/1/1$  case. Thus,  $\delta = \lambda$  in this case.

**s.5.4.11.** It is evident that the expected waiting time for an arriving customer is the expected remaining service time plus the expected time in queue. The expected time in queue must be equal to the expected number of customers in queue at an arrival epoch times the expected service time per customer, assuming that service times are i.i.d. If the arrival process is Poisson, it follows from PASTA that the average number of jobs in queue perceived by arriving customers is also the *time-average* number of jobs in queue  $E[L_Q]$ .

**s.5.4.12.** With Little's law  $E[L_Q] = \lambda E[W_Q]$ . Using this,

$$E[W_Q] = E[S_r] + \lambda E[W_Q] E[S] = E[S_r] + \rho E[W_Q],$$

since  $\rho = \lambda E[S]$ . But this gives for the  $M/G/1$  queue that

$$E[W_Q] = \frac{E[S_r]}{1 - \rho}.$$

**s.5.4.13.** Observe that when  $s \in [\tilde{A}_k, D_k)$ , the remaining service time until job  $k$  departs is  $D_k - s$ , while if  $s \notin [\tilde{A}_k, D_k)$ , job  $k$  is not in service so it cannot have any remaining service.

At time  $s$ , the number of departures is  $D(s)$ . Thus,  $D(s) + 1$  is the first job to depart after time  $s$ . The departure time of this job is  $D_{D(s)+1}$ , hence the remaining service time at time  $s$  is  $D_{D(s)+1} - s$ , provided this job is in service.

**s.5.4.14.** It is clear that the time-average  $Y(t)/t \rightarrow E[S_r]$ . Moreover,  $X_k = Y(D_k) - Y(D_{k-1})$  is the area under the triangle in Fig. 15. Thus,  $X = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n S_k^2/2 = E[S^2]/2$ . Finally,  $\delta = \lambda$  by rate-stability.

The rest is plain algebra, see [5.4.15].

**s.5.4.15.** Use that

$$\frac{E[S^2]}{(E[S])^2} = \frac{(E[S^2] - (E[S])^2) + (E[S])^2}{(E[S])^2} = \frac{V[S] + (E[S])^2}{(E[S])^2} = C_s^2 + 1.$$

Then

$$\lambda E[S^2] = \frac{E[S^2]}{(E[S])^2} \lambda (E[S])^2 = \frac{E[S^2]}{(E[S])^2} \rho E[S] = (1 + C_s^2) \rho E[S].$$

**s.5.4.16.** By the PASTA property we know that  $\rho$  is the probability to find the server busy upon arrival; hence  $1 - \rho$  is the probability that the server is idle upon arrival. Then,

$$E[S_r] = \rho E[S_r | S_r > 0] + (1 - \rho) E[S_r | S_r = 0] = \rho E[S_r | S_r > 0],$$

since, evidently,  $E[S_r | S_r = 0] = 0$ .

**s.5.4.17.**  $E[S_r | S_r > 0] = E[S]$  for the  $M/M/1$  queue, and  $E[S_r] = \rho E[S_r | S_r > 0]$  for the  $M/G/1$  queue, it follows that

$$E[S_r] = \rho E[S_r | S_r > 0] = \rho E[S].$$

**s.5.4.18.** From (5.4.5) and the sentences above this equation, we have that

$$E[S_r] = \rho E[S_r | S_r > 0] = \lambda E[S] E[S_r | S_r > 0].$$

Hence,

$$\lambda E[S_r | S_r > 0] = \frac{E[S_r]}{E[S]} = \lambda \frac{E[S^2]}{2E[S]}.$$

**s.5.4.19.** When  $S$  is deterministic  $E[S^2] = (E[S])^2$ . Now use (5.4.6) to see that it is true.

**s.5.4.20.** By [5.4.19], it is half the service time.

**s.5.4.21.** When  $S$  is uniform on  $[0, \alpha]$ ,

$$E[S] = \alpha^{-1} \int_0^\alpha x \, dx = \frac{\alpha^2}{2\alpha} = \frac{\alpha}{2}, \quad E[S^2] = \alpha^{-1} \int_0^\alpha x^2 \, dx = \frac{\alpha^2}{3}.$$

Thus, with (5.4.6),  $E[S_r | S_r > 0] = (\alpha^2/3)/(2\alpha/2) = \alpha/3$ .

**s.5.4.22.**

$$\begin{aligned} E[S] &= \mu \int_0^\infty x e^{-\mu x} \, dx = 1/\mu, \\ E[S^2] &= \mu \int_0^\infty x^2 e^{-\mu x} \, dx = -x^2 e^{-\mu x} \Big|_0^\infty + 2 \int_0^\infty x e^{-\mu x} \, dx \\ &= -2 \frac{x}{\mu} e^{-\mu x} \Big|_0^\infty + \frac{2}{\mu} \int_0^\infty e^{-\mu x} \, dx = \frac{2}{\mu^2}. \end{aligned}$$

Now use (5.4.6).

## 5.5 $M^X/M/1$ QUEUE LENGTH DISTRIBUTION

In Sections 5.3 and 5.4 we established the Pollaczek-Khinchine formula for the waiting times of the  $M^X/M/1$  queue and the  $M/G/1$  queue, respectively. To compute more difficult performance measures such as the loss probability  $P(L > n)$ , we need expressions for the stationary distribution  $\pi(n)$ . Here we present a numerical, recursive, scheme to compute these probabilities. Recall, that by the PASTA property, the long-run fraction of time  $p(n)$  the system contains  $n$  jobs is the same as the fraction  $\pi(n)$  of arriving jobs that observe  $n$  jobs in the system. Hence, by finding  $\pi(n)$  we also obtain  $p(n)$ .

### *Theory and Exercises*

To find  $\pi(n)$ ,  $n = 0, 1, \dots$ , we turn again to level-crossing arguments. However, the reasoning that led to the level-crossing equation (4.2.4) needs to be generalized. To see this, we consider an example. If  $L(t) = 3$ , the system contains 3 items. (This is not necessarily the same as 3 batches.) Since the server serves single items, down-crossings of level  $n = 3$  occur in single units.

However, due to the batch arrivals, when a job arrives it typically brings multiple items to the queue. For instance, suppose that  $L(A_k-) = 3$ , i.e., job  $k$  sees 3 items in the system at its arrival epoch. If its size  $B_k = 20$ , then right after the  $k$ th arrival the system contains 23 items, that is,  $L(A_k) = 3 + 20 = 23$ . Thus, upon the arrival of job  $k$ , all levels between states 3 and 23 are crossed.

The left panel in Fig. 16 shows all up- and down-crossings of some level  $n$ . The down-crossing rate is easy: just as in Fig. 10 there is just one arrow from right to left. However, level  $n$  can be up-crossed from below from many states, in fact from any level  $m \in \{0, 1, \dots, n-1\}$ . More formally, to count the number of up-crossings define

$$A(m, n, t) = \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-) = m} \mathbb{1}_{B_k > n-m}$$

as the number of jobs up to time  $t$  that see  $m$  in the system upon arrival and have batch size larger than  $n - m$ . Recall from Section 5.3 that  $f(k) = P(B = k)$  and  $G(k) = P(B > k)$ .

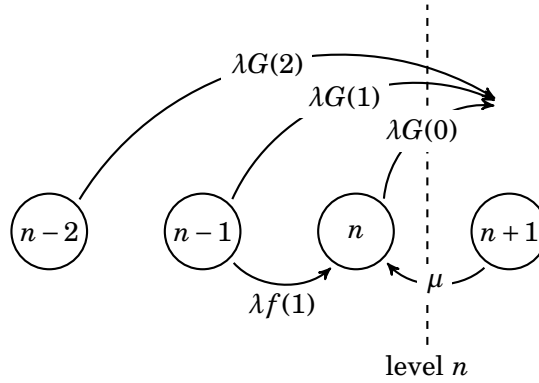


Figure 16: Level crossing of level  $n$ . Observe that when the system is in state  $n-2$ , the arrival of any batch larger than 2 ensures that level  $n$  is crossed from below. The rate at which such events happen is  $\lambda\pi(n-2)G(2)$ . Similarly, in state  $n-1$ , the arrival of any batch larger than one item ensures that level  $n$  is crossed, and this occurs with rate  $\lambda\pi(n-1)G(1)$ , and so on.

**5.5.1.** Show that  $A(n, n, t) = A(n, t)$ , where  $A(n, t)$  is defined by (4.2.1a).

**5.5.2.** Provided the limit exists, show that

$$\lim_{t \rightarrow \infty} \frac{A(m, n, t)}{A(m, t)} = P(B > n - m | L(A-) = m), \quad (5.5.1)$$

where the random variable  $L(A-)$  denotes the number in the system seen by an arbitrary arrival.

**5.5.3.** Show that  $P(B > n - m | L(A-) = m) = P(B > n - m)$ .

**5.5.4.** Assuming that the limits exist, show that

$$\lim_{t \rightarrow \infty} \frac{A(m, n, t)}{t} = \lambda\pi(m)G(n - m).$$

Equating the number of up- and down-crossing gives  $\sum_{m=0}^n A(m, n, t) \approx D(n, t)$ . Then, dividing by  $t$ , taking the limit  $t \rightarrow \infty$ , and using [5.5.4] results in the level-crossing equation for the  $M^X/M/1$  queue:

$$\lambda \sum_{m=0}^n \pi(m)G(n - m) = \mu\pi(n + 1). \quad (5.5.2)$$

**5.5.5.** Provide an interpretation of (5.5.2) in terms of a thinned Poisson arrival process.

**5.5.6.** Show that (5.5.2) reduces to  $\lambda\pi(n) = \mu\pi(n+1)$  for the  $M/M/1$  case.

It is left to find the normalization constant. As the recursion (5.5.2) does not lead to a closed form expression for  $\pi(n)$ , such as (5.1.1), we need to use a criterion to stop this iterative procedure. It is not so easy to find general conditions when to stop, but we can use a pragmatic approach. When the demand is finite, the numbers  $\{\pi(k)\}$  should decrease geometrically fast for all  $k \geq N$  where  $N$  is some large number<sup>1</sup>. Stop when  $\pi(N) \ll \pi(0)$ , and take  $\sum_{i=0}^N \pi(i)$  as the normalization constant.

Once we have  $\pi(n)$ , we can compute the influence on the batch size distribution,  $\lambda$ , and  $\mu$  on the system's performance.

**5.5.7.** Why is (4.3.7), i.e.,  $\pi(n) = \delta(n)$ , not true for the  $M^X/M/1$  batch queue? Provide an example.

**5.5.8.** Show that

$$\mu E[L] = \mu \sum_{n=0}^{\infty} n\pi(n) = \lambda \frac{E[B^2]}{2} + \lambda E[B] E[L] + \lambda \frac{E[B]}{2}. \quad (5.5.3)$$

**5.5.9.** Use (5.5.3) and the definition  $\rho = \lambda E[B]/\mu$  to show that

$$(1 - \rho)E[L] = \frac{\lambda}{\mu} \frac{E[B^2]}{2} + \frac{\rho}{2}.$$

Observe that this is in line with [5.3.7].

**5.5.10.** Substitute recursion (5.5.2) for  $\pi(n)$  into the expression  $E[L] = \sum_{n=0}^{\infty} n\pi(n)$  and derive (5.3.2).

**5.5.11.** Implement the recursion (5.5.2) in a computer program for the case  $f(1) = f(2) = f(3) = 1/3$ . Take  $\lambda = 1$  and  $\mu = 3$ .

We consider the  $M^X/M/1/K$  queue, i.e., a batch queue in which at most  $K$  jobs fit into the system. When customers can be blocked in a batch queue it is necessary to specify a policy that decides which items in a batch to accept. Three common rules are

1. Complete rejection: if a batch does not fit entirely into the system, it will be rejected completely.
2. Partial acceptance: accept whatever fits of a batch, and reject the rest.
3. Complete acceptance: accept all batches that arrive when the system contains  $K$  or less jobs, and reject the entire batch otherwise.

**5.5.12.** Derive a set of recursions, analogous to (5.5.2), to compute  $\pi(n)$  for the  $M^X/M/1/K$  queue with complete rejection.

**5.5.13.** Derive a set of recursions, analogous to (5.5.2), to compute  $\pi(n)$  for the  $M^X/M/1/K$  queue with complete acceptance.

**5.5.14.** Derive a set of recursions, analogous to (5.5.2), to compute  $\pi(n)$  for the  $M^X/M/1/K$  queue with partial acceptance.

**5.5.15** (Batch services, not obligatory). An interesting extension is a queueing process with batch services, i.e., the  $M/M^Y/1$  queue. Constructing a recursion for the steady-state probabilities  $\pi(n)$  for this case is not hard, in fact, mostly analogous to (5.5.2). However, solving the recursion appears to be quite a bit harder.

<sup>1</sup> An interesting question, why should it decrease monotonically after some, large,  $N$ ?

*Hints*

**h.5.5.3.** Realize that  $B$  and  $L(A-)$  are assumed to be independent.

**h.5.5.8.** Substitute the recursion, and carry on with the algebra. Use also the results of the exercises of Section 5.3.

*Solutions*

**s.5.5.1.**  $A(n, n, t)$  counts all jobs up to time  $t$  that see  $n$  items and bring at least  $n - n + 1 = 1$  unit of work. As each job brings at least 1 item,  $A(n, n, t)$  counts all jobs that see  $n$  items at arrival.

**s.5.5.2.**

$$\lim_{t \rightarrow \infty} \frac{A(m, n, t)}{A(m, t)} = \lim_{t \rightarrow \infty} \frac{\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-) = m, B_k > n-m}}{\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-) = m}} = P(B > n - m | L(A-) = m),$$

**s.5.5.3.**

$$\begin{aligned} P(B > n - m | L(A-) = m) &= \frac{P(B > n - m, L(A-) = m)}{P(L(A-) = m)} \\ &= \frac{P(B > n - m) P(L(A-) = m)}{P(L(A-) = m)} \\ &= P(B > n - m) = G(n - m). \end{aligned}$$

**s.5.5.4.** For this purpose, observe that we can write

$$\frac{A(m, n, t)}{t} = \frac{A(t)}{t} \frac{A(m, t)}{A(t)} \frac{A(m, n, t)}{A(m, t)}. \quad (5.5.4)$$

By the assumptions of Section 4.3,  $A(t)/t \rightarrow \lambda$  and  $A(m, t)/A(t) \rightarrow \pi(m)$ . Next, apply the results of [5.5.2] to [5.5.3].

**s.5.5.5.** (5.5.4) has the interpretation that the rate at which level  $n$  is crossed from below from state  $m$  is equal to the rate at which jobs arrive times the fraction of jobs that see  $m$  jobs in the system times the fraction of jobs with batch size larger than  $n - m$ . Observe that the stream of jobs with batch size larger than  $n - m$  is a Poisson process thinned at rate  $G(n - m)$ .

**s.5.5.6.** The right-hand side of (5.5.2) is identical, so we only have to concentrate on the left-hand side. In the  $M/M/1$  queue, all batches have size 1. Thus,  $P(B = 1) = f(1) = 1$  and  $f(k) = 0$  for  $k \neq 1$ . Thus,  $G(0) = 1$  and  $G(1) = G(2) = \dots = 0$ . Thus,  $\sum_{m=0}^n G(n - m) \pi(m) = G(0) \pi(n) = \pi(n)$ .

**s.5.5.7.** Using [4.3.5] that  $D(n, t)/t \rightarrow \delta \delta(n)$ , we see that  $\lambda \sum_{m=0}^n \pi(m) G(n - m) = \delta \delta(n)$ . With rate-stability,  $\lambda = \delta$ , and therefore  $\sum_{m=0}^n \pi(m) G(n - m) = \delta(n)$ . The left-hand-side is clearly not equal to  $\pi(n)$ .

**s.5.5.8.** We use that  $\mu \pi(n) = \lambda \sum_{i=0}^{n-1} \pi(i) G(n - 1 - i)$  and the results of the exercises of Section 5.3 to see that

$$\mu E[L] = \sum_{n=0}^{\infty} n \mu \pi(n), \quad \text{now substitute for } \mu \pi(n) \text{ the recursion (5.5.2),}$$

$$\begin{aligned}
&= \lambda \sum_{n=0}^{\infty} n \sum_{i=0}^{n-1} \pi(i) G(n-1-i) = \lambda \sum_{n=0}^{\infty} n \sum_{i=0}^{\infty} \mathbb{1}_{i < n} \pi(i) G(n-1-i) \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} \mathbb{1}_{i < n} n G(n-1-i) = \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=i+1}^{\infty} n G(n-1-i) \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} (n+i+1) G(n) = \lambda \sum_{i=0}^{\infty} \pi(i) \left[ \sum_{n=0}^{\infty} n G(n) + (i+1) \sum_{n=0}^{\infty} G(n) \right] \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} n G(n) + \lambda E[B] \sum_{i=0}^{\infty} \pi(i) (i+1) \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \frac{E[B^2] - E[B]}{2} + \lambda E[B] (E[L] + 1) \\
&= \lambda \frac{E[B^2] - E[B]}{2} + \lambda E[B] E[L] + \lambda E[B] \\
&= \lambda \frac{E[B^2]}{2} + \lambda E[B] E[L] + \lambda \frac{E[B]}{2}.
\end{aligned}$$

**s.5.5.9.** Dividing both sides of (5.5.3) by  $\mu$  and using that  $\lambda E[B]/\mu = \rho$ ,

$$E[L] = \frac{\lambda}{\mu} \frac{E[B^2]}{2} + \rho E[L] + \frac{\rho}{2}.$$

**s.5.5.10.** We chop this up into two steps, see [5.5.8] and [5.5.9].

**s.5.5.11.** The recursion for  $n$  becomes

$$\pi(n) = \frac{\lambda}{\mu} \sum_{i=0}^{n-1} \pi(n-1-i) G(i).$$

Since  $G(i) = 0$  for  $i \geq 3$  we rewrite this to

$$\pi(n) = \frac{\lambda}{\mu} \sum_{i=0}^{\min\{n-1, \text{len } G\}} \pi(n-1-i) G(i),$$

where  $\text{len } G$  is the largest possible batch size.

The following code carries out the recursion for the  $M/M/1$  queue and compares the result to the *unnormalized* probabilities of the  $M/M/1$  queue. Recall,  $(\lambda/\mu)^n$  are these unnormalized probabilities. With this we can test the code right away.

```

>>> import numpy as np

>>> f = [0, 1] # set f[0] = 0
>>> F = np.cumsum(f) # distribution
>>> G = np.ones_like(F) - F # survivor function

>>> labda = 1
>>> mu = 3
>>> num = 4
>>> p = np.ones(num)

>>> for n in range(1, num):

```

```

...     p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, len(G))))
...     print(n, p[n], (labda / mu) ** n)
...
1 0.3333333333333333 0.3333333333333333
2 0.1111111111111111 0.1111111111111111
3 0.037037037037037035 0.03703703703703703
>>> #

```

The test is convincing. So now we'll move on the real problem.

```

>>> f = np.ones(4) / 3
>>> f[0] = 0
>>> F = np.cumsum(f)
>>> G = np.ones_like(F) - F
>>> num = 30 # stop at 30
>>> p = np.ones(num)
>>> for n in range(1, num):
...     p[n] = sum(p[n - 1 - m] * G[m] for m in range(min(n, len(G))))
...
>>> p *= labda / mu
>>> p /= p.sum() # normalize

```

Here we use a small trick to speed up the computations. Note that  $\sum_{m=0}^n \pi(m)G(n-m) = \sum_{m=0}^n \pi(n-m)G(m)$ . We also know that  $G(m) = 0$  for  $m$  sufficiently large; here this happens for sure when  $m \geq \text{len}(G)$ . Thus, we don't have to evaluate the summation up to  $n$ , but we can stop once we reach  $\text{len}(G)$ . Another trick is to take the multiplication  $\lambda/\mu$  out of the for-loop.

Now that we have the probabilities we can do all experiments we like.

```

>>> EL = sum(n * p[n] for n in range(len(p)))
>>> EL
27.2237869364617

```

It is of interest to compare this result to (5.3.2). In fact, I spent at least one hour to get (5.3.2) correct; now I can finally check it numerically.

```

>>> EB = sum(k * fk for k, fk in enumerate(f))
>>> rho = labda * EB / mu
>>> EB2 = sum(k * k * fk for k, fk in enumerate(f))
>>> VB = EB2 - EB * EB
>>> C2 = VB / EB / EB
>>> EL = (1 + C2) / 2 * rho / (1 - rho) * EB + rho / (1 - rho) / 2
>>> EL
3.3333333333333326

```

So, after all, stopping at 30 is not quite OK: there is a slight difference between the two expectations. Let's run the recursion up to 100, and see what we get then.

```

>>> num = 100
>>> p = np.ones(num)

```

```

>>> for n in range(1, num):
...     p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, len(G))))
...
>>> p /= p.sum() # normalize
>>> EL = sum(n * p[n] for n in range(len(p)))
>>> print(EL)
3.3333333271183934

```

This does the job.

**s.5.5.12.** Suppose a batch of size  $k$  arrives when the system contains  $n$  jobs. When  $k + n \leq K$ , the batch can be accepted since the entire batch will fit into the queue. When, however,  $k + n > K$ , the batch has to be rejected.

Now consider an imaginary line between states with  $n$  and  $n + 1$  jobs in the system. This imaginary line separates the state space into two disjoint parts: one part with states  $0, 1, \dots, n$  and the other part with states  $n + 1, n + 2, \dots, K$ . We call this imaginary line ‘level  $n$ ’. We call an ‘up-crossing’ a transition from some state  $m \leq n$  to a state  $l > n$ . Likewise, a ‘down-crossing’ of level  $n$  is a transition from some state  $l > n$  to some state  $m \leq n$ .

If the system contains  $n$  jobs, level  $n$  is crossed from below with rate  $\lambda\pi(n)P(B \leq K - n)$ . More generally, when the system is in state  $m \leq n$ , we need a batch of at least  $n + 1 - m$  to cross level  $n$ . Moreover, any batch larger than  $K - m$  gets rejected. Thus, when the system contains  $m \leq n$  jobs, the rate at which level  $n$  is crossed from below is

$$\lambda\pi(m)P(n + 1 - m \leq B \leq K - m) = \lambda\pi(m)[G(n - m) - G(K - m)],$$

where we use that

$$\begin{aligned}
 P(n + 1 - m \leq B \leq K - m) &= P(B \leq K - m) - P(B \leq n - m) \\
 &= P(B > n - m) - P(B > K - m) \\
 &= G(n - m) - G(K - m).
 \end{aligned}$$

Since the server serves only single items, level  $n$  can only be crossed from above from state  $n + 1$ . This happens at rate  $\mu p(n + 1)$ . With PASTA this is equal to  $\mu\pi(n + 1)$ .

Finally, since in the long run the number of up- and down-crossings must be the same, the up- and down-crossing rates must match. This implies that the balance equations becomes

$$\mu\pi(n + 1) = \lambda \sum_{m=0}^n \pi(m)[G(n - m) - G(K - m)],$$

for  $n = 0, \dots, K - 1$ .

Before we continue with the other acceptance rules, it is important to check this result. In general, off-by-one errors are easily, and commonly, made, so we need to test the above on simple cases.

- If  $K \rightarrow \infty$ , then  $G(K - m) = P(B > K - m) \rightarrow 0$ , so we get our earlier result.
- Take  $n = K$ . Then  $G(n - m) - G(K - m) = 0$  for all  $m$ . Then the right-hand side is 0, as it should.



- Taken  $n = 0$  and  $K = 1$ , then  $\mu\pi(1) = \lambda\pi(0)$ . This also makes sense.
- Take  $n$  much smaller than  $K$ . If the batch size is maximally 2, then for small  $n$  the entire batch must fit. Let's see if this holds in the above formula. If  $n$  much smaller than  $K$ , then also  $m$  is much smaller than  $K$  (since in the right-hand side,  $m \leq n$ ). But then  $G(K - m) \leq G(2) = 0$ , as it should. (Observe that  $G$  is a decreasing function of its argument; it's a survival function.)

**s.5.5.13.** The complete-acceptance policy is actually quite simple. As any batch will be accepted when  $n \leq K$ , the queue length is not bounded. Only when the number of jobs in the system is larger than  $K$ , we do not accept jobs.

$$\mu\pi(n+1) = \begin{cases} \lambda \sum_{m=0}^n \pi(m)G(n-m), & \text{for } n \leq K, \\ \lambda \sum_{m=0}^K \pi(m)G(n-m), & \text{for } n > K. \end{cases}$$

**s.5.5.14.** For the partial acceptance case, any job is accepted, but the system only admits whatever fits. As level  $n \in 0, 1, \dots, K-1$  is still up-crossed by any batch of size at least  $n-m$  when the system is in state  $m$ , the formula for the up-crossing rate is identical to the case without this acceptance policy. Moreover, nothing changes to the formula for the down-crossing rate. Hence,

$$\mu\pi(n+1) = \lambda \sum_{m=0}^n \pi(m)G(n-m),$$

for  $n = 0, 1, \dots, K-1$ .

## 5.6 M/G/1 QUEUE LENGTH DISTRIBUTION

In Section 5.5 we used level-crossing arguments to find a recursive method to compute the stationary distribution  $p(n)$  of the number of items in an  $M^X/M/1$  queue. Here we apply similar arguments to find  $p(n) = P(L = n)$  for the  $M/G/1$  queue. However, we cannot simply copy the derivation of the  $M^X/M/1$  queue to the  $M/G/1$  queue, because in the  $M^X/M/1$  queue the service times of the items are exponential, hence memoryless, while in the  $M/G/1$  this is not the case.

### Theory and Exercises

When job service times are not memoryless, hence do not restart at arrival times, we cannot choose any moment we like to apply level-crossing. Thus, for the  $M/G/1$  queue we need to focus on moments in time in which the system 'restarts'. As we will see below, the appropriate moments are job departure epochs. All in all, the argumentation to find the recursion for  $\{p(n)\}$  is quite subtle, as it uses an interplay of the PASTA property and (4.3.7) between  $\pi(n)$ ,  $p(n)$  and  $\delta(n)$ .

An important role below is played by the number of arrivals  $Y_k$  during the service time of the  $k$ th job. Since the service times of the jobs form a sequence of i.i.d. random variables, the elements of the sequence  $\{Y_k\}$  are also i.i.d. Let  $Y$  be the common random variable with probability mass  $f(j) = P(Y = j)$ ; write  $G(j) = P(Y_k > j)$  for the survivor function.

**5.6.1.** Explain that if the service time is constant and equal to  $s$ , then

$$P(Y_k = j | S = s) = e^{-\lambda s} \frac{(\lambda s)^j}{j!}. \quad (5.6.1)$$

**5.6.2.** Explain that

$$P(Y_k = j) = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!} dF(x), \quad (5.6.2)$$

where  $F$  is the distribution of the service times.

**5.6.3.** If  $S$  is deterministic and equal to  $s$ , show that (5.6.2) reduces to (5.6.1).

**5.6.4.** If  $S \sim \text{Exp}(\mu)$ , show that

$$f(j) = P(Y_k = j) = \frac{\mu}{\lambda + \mu} \left( \frac{\lambda}{\lambda + \mu} \right)^j. \quad (5.6.3)$$

**5.6.5.** If  $S \sim \text{Exp}(\mu)$ , show that

$$G(j) = \sum_{k=j+1}^{\infty} f(k) = \left( \frac{\lambda}{\lambda + \mu} \right)^{j+1}. \quad (5.6.4)$$

**5.6.6.** Design a suitable numerical method to evaluate (5.6.2) for more general distribution functions  $F$ .

**5.6.7.** Show that  $E[Y] = \lambda E[S] = \rho$ .

Let us concentrate on a down-crossing of level  $n$ , see Fig. 17; recall that level  $n$  lies between states  $n$  and  $n+1$ . For job  $k$  to generate a down-crossing of level  $n$ , two events must take place: job ' $k-1$ ' must leave  $n+1$  jobs behind after its service completion, and job  $k$  must leave  $n$  jobs behind. Thus,

$$\text{Down-crossing of level } n \iff \mathbb{1}_{L(D_{k-1})=n+1} \mathbb{1}_{L(D_k)=n} = 1.$$

Let us write this in another way. Observe that if  $L(D_{k-1}) = n+1$  and no other jobs arrive during the service time  $S_k$  of job  $k$ , i.e., when  $Y_k = 0$ , it must also be that job  $k$  leaves  $n$  jobs behind. If, however,  $Y_k > 0$ , then  $L(D_k) \geq n+1$ . Thus, we see that

$$\text{Down-crossing of level } n \iff \mathbb{1}_{L(D_{k-1})=n+1} \mathbb{1}_{Y_k=0} = 1.$$

Consequently, the number of down-crossings of level  $n$  up to time  $t$  is

$$D(n+1, 0, t) = \sum_{k=1}^{D(t)} \mathbb{1}_{L(D_{k-1})=n+1} \mathbb{1}_{Y_k=0}.$$

**5.6.8.** Show that

$$\lim_{t \rightarrow \infty} \frac{D(n+1, 0, t)}{t} = \delta \delta(n+1) f(0),$$

where  $f(0) = P(Y = 0)$ .

Before we deal with the up-crossing, it is important to do the next exercise.

**5.6.9.** Suppose that  $L(D_{k-1}) > 0$ . Why is  $D_k = D_{k-1} + S_k$ ? However, if  $L(D_{k-1}) = 0$ , the time between  $D_{k-1}$  and  $D_k$  is not equal to  $S_k$ . Why not? Can you find an expression for the distribution of  $D_k - D_{k-1}$  in case  $L(D_{k-1}) = 0$ ?

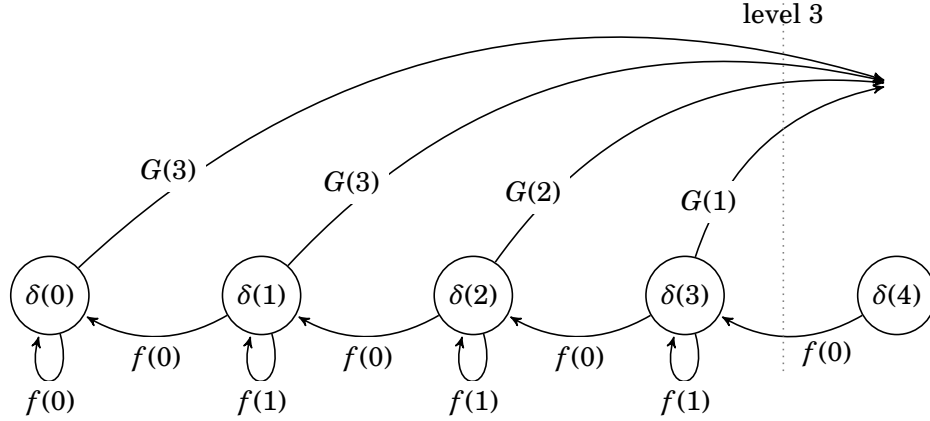


Figure 17: Level 3 is crossed from below with rate  $\delta\delta(0)G(3) + \delta\delta(1)G(3) + \dots + \delta\delta(3)G(1)$  and crossed from above with rate  $\delta\delta(4)f(0)$ .

For the up-crossings, assume first that  $L(D_{k-1}) = n > 0$ . Then an up-crossing of level  $n > 0$  must have occurred when  $L(D_k) > n$ , i.e.,

$$\mathbb{1}_{L(D_{k-1})=n} \mathbb{1}_{L(D_k)>n} = 1 \implies \text{Up-crossing of level } n.$$

Again, we can convert this into a statement about the number of arrivals  $Y_k$  that occurred during the service time  $S_k$  of job  $k$ . If  $Y_k = 0$ , then job  $k$  must leave  $n - 1$  jobs behind, so no up-crossing can happen. Next, if  $Y_k = 1$ , then job  $k$  leaves  $n$  jobs behind, so still no up-crossing occurs. In fact, level  $n$  can only be up-crossed from level  $n$  if more than one job arrives during the service of job  $k$ , i.e.,

$$\mathbb{1}_{L(D_{k-1})=n} \mathbb{1}_{Y_k>1} = 1 \implies \text{Up-crossing of level } n.$$

More generally, level  $n$  is up-crossed from level  $m$ ,  $0 < m \leq n$  whenever

$$\mathbb{1}_{L(D_{k-1})=m} \mathbb{1}_{Y_k>n-m+1} = 1 \implies \text{Up-crossing of level } n.$$

However, if  $m = 0$  (think about this),

$$\mathbb{1}_{L(D_{k-1})=0} \mathbb{1}_{Y_k>n} = 1 \implies \text{Up-crossing of level } n.$$

Again we define proper counting functions, divide by  $t$ , and take suitable limits to find for the up-crossing rate

$$\delta\delta(0)G(n) + \delta \sum_{m=1}^n \delta(m)G(n-m+1). \quad (5.6.5)$$

Equating the down-crossing and up-crossing rates and dividing by  $\delta$  gives

$$f(0)\delta(n+1) = \delta(0)G(n) + \sum_{m=1}^n \delta(m)G(n+1-m).$$

Noting that  $\pi(n) = \delta(n)$ , as this holds for any rate-stable G/G/1 queue, cf., (4.3.7), hence in particular for the M/G/1 queue length process, we arrive at

$$f(0)\pi(n+1) = \pi(0)G(n) + \sum_{m=1}^n \pi(m)G(n+1-m). \quad (5.6.6)$$

Clearly, we have again obtained a recursion with which we can compute the state probabilities.

**5.6.10.** Provide the details behind the derivation of (5.6.5).

**5.6.11.** Clearly, the  $M/M/1$  queue is a special case of the  $M/G/1$  queue. Check that the queue length distribution of the  $M/M/1$  queue satisfies (5.6.6).

*Hints*

**h.5.6.1.** If  $s$  is deterministic, the number of arrivals during a fixed period of time with length  $s$  must be Poisson distributed.

**h.5.6.2.** Use [5.6.1].

**h.5.6.3.**  $S = s$  then  $P(S = s) = 1$ , i.e., all probability mass lies at  $s$ . Thus, all arrivals must occur during  $[0, s]$ .

**h.5.6.4.** Use the ideas of [1.3.3] to simplify the standard integral.

**h.5.6.5.** Use [5.6.4].

**h.5.6.6.** Discretize time to a grid of points, and approximate the integral by a summation over the grid.

**h.5.6.8.** Use a similar derivation as in (5.5.1).

**h.5.6.9.** Realize that if  $L(D_{k-1}) = 0$ , job  $k-1$  leaves behind an empty system. Thus, before job  $k$  can leave, it has to arrive. In other words,  $D_{k-1} < A_k$ . Since job  $k$  arrives to an empty system, his service starts right away, so that the time between  $A_k$  and  $D_k$  is equal to the service time of job  $k$ .

**h.5.6.10.** Define for  $m = 1, \dots, n$

$$D(m, n, t) = \sum_{k=1}^{D(t)} \mathbb{1}_{L(D_{k-1})=m} \mathbb{1}_{Y_k > n-m+1},$$

and

$$D(0, n, t) = \sum_{k=1}^{D(t)} \mathbb{1}_{L(D_{k-1})=0} \mathbb{1}_{Y_k > n}.$$

Then, divide by  $D(m, t)$  and  $D(t)$  and take limits.

**h.5.6.11.** To simplify the computations, define shorthands such as  $\alpha = \lambda/(\lambda + \mu)$ , so that  $1 - \alpha = \mu/(\lambda + \mu)$ , and  $\alpha/(1 - \alpha) = \lambda/\mu = \rho$ . Then, with (5.6.3),  $f(n) = \alpha^n(1 - \alpha)$  and  $G(n) = \alpha^{n+1}$ . (These results do not 'come for free'. Of course, it's just algebra, but please try to derive this yourself. It's good to hone your computational skills.)

### Solutions

**s.5.6.1.** See the hint. The period during which the arrivals occur is  $s$ .

**s.5.6.2.** We use a conditioning argument to arrive at this result. The probability that the service time is  $x$  units long is written in various ways in the literature:  $F(dx) = dF(x) = P(S \in dx)$ , but this all means the same thing, it is only the notation that differs. (As an aside, to properly define this we need measure theory). When  $F$  has a density  $f$ , then  $dF(x) = f(x)dx$ . Note that when  $S$  is discrete, it does not have a density everywhere. With this,

$$P(Y_k = j) = \int_0^\infty P(Y_k = j | S = x) P(S \in dx) = \int_0^\infty P(Y_k = j | S = x) dF(x).$$

Using the answer of the previous problem we arrive at the result.

**s.5.6.3.** Let us first attack this problem from a general point of view. Suppose the service time  $S$  can take values  $s_1 < s_2 < \dots < s_n$ , and  $P(S = s_i) = \alpha_i$ . Then of course we want that  $P(S \leq s_n) = \sum_{i=1}^n \alpha_i = 1$ . The distribution function  $F$  of  $S$  is in this case a step function, with steps at the points  $s_1, s_2, \dots$ , and step sizes  $\alpha_1, \alpha_2, \dots$ . Thus,  $F(s_i) - F(s_i-) = \alpha_i$ . We say that such a distribution function has *atoms* at the points  $s_1, s_2, \dots$ . In this case we write

$$\int_0^\infty g(x) dF(x) = \sum_{i=1}^n g(s_i) \alpha_i.$$

Thus, the integral of  $g$  with respect to  $F$  is the sum of  $g$  at the points at which  $F$  makes a jump times the weight of  $F$  at these points.

As an example, in case  $S \equiv 10$  (the service time is always 10 time units long) the distribution function  $F$  makes just one jump at  $s_1 = 10$  of size  $\alpha_1 = F(10) - F(10-) = 1$ , i.e.,  $F$  has the form

$$F(x) = \begin{cases} 0, & x < 10, \\ 1, & x \geq 10. \end{cases}$$

With this,

$$\int_0^\infty g(x) dF(x) = \sum_{i=1}^n g(s_i) \alpha_i = g(s_1) \alpha_1 = g(10) \cdot 1 = g(10).$$

Thus, the integral of  $g$  with respect to this distribution  $F$  is  $g(10)$ . More generally, when  $F$  puts all probability mass at the single point  $s$  (rather than at the point 10), then

$$\int_0^\infty g(x) dF(x) = g(s).$$

Let us now copy the formula of the previous problem:

$$P(Y_k = j) = \int_0^\infty P(Y_k = j | S = x) dF(x).$$

We see that here the integrand is the function  $g(x) = P(Y_k = j | S = x)$ . It is given in the question that  $F$  puts all mass on the point  $s$ . Therefore,

$$P(Y_k = j) = \int_0^\infty P(Y_k = j | S = x) dF(x) = \int_0^\infty g(x) dF(x) = g(s). \quad \star$$

Now we also know that if the service time takes precisely  $x$  time units, the number of arrivals is Poisson distributed. Therefore

$$P(Y_k = j | S = x) = e^{-\lambda x} \frac{(\lambda x)^j}{j!}.$$

Hence,

$$g(x) = P(Y_k = j | S = x) = e^{-\lambda x} \frac{(\lambda x)^j}{j!}.$$

Finally, using  $(\star)$ , and taking  $x = s$  in the above expression of  $g$ , we get

$$P(Y_k = j) = \int_0^\infty P(Y_k = j | S = x) dF(x) = g(s) = e^{-\lambda s} \frac{(\lambda s)^j}{j!}.$$

**s.5.6.4.** Use conditional probability to see that

$$\begin{aligned} P(Y_n = j) &= \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!} dF(x) = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!} \mu e^{-\mu x} dx \\ &= \frac{\mu}{j!} \lambda^j \int_0^\infty e^{-(\lambda+\mu)x} x^j dx = \frac{\mu}{j!} \left( \frac{\lambda}{\lambda+\mu} \right)^j \int_0^\infty e^{-(\lambda+\mu)x} ((\lambda+\mu)x)^j dx \\ &= \frac{\mu}{j!} \left( \frac{\lambda}{\lambda+\mu} \right)^j \frac{j!}{\lambda+\mu}. \end{aligned}$$

In the last integral, we use the hint. Specifically,

$$\begin{aligned} \int_0^\infty e^{-\alpha x} (\alpha x)^j dx &= \alpha^{-1} \int_0^\infty e^{-x} x^j dx = \alpha^{-1} \left( -e^{-x} x^j \Big|_0^\infty + j \int_0^\infty e^{-x} x^{j-1} dx \right) \\ &= \alpha^{-1} j \int_0^\infty e^{-x} x^{j-1} dx = \alpha^{-1} j(j-1) \int_0^\infty e^{-x} x^{j-2} dx \\ &= \alpha^{-1} j! \int_0^\infty e^{-x} dx = j!/\alpha. \end{aligned}$$

In hindsight, this result could have been derived in another way, in fact by using the result of [1.3.16] in which we analyzed a merged Poisson process. Consider the Poisson process with rate  $\lambda + \mu$  that arises when the arrival and service process are merged. The probability that an arrival corresponds to an epoch of the merged process is  $\lambda/(\lambda + \mu)$  and the probability that a departure corresponds to an epoch of the merged process is  $\mu/(\lambda + \mu)$ . The probability that  $j$  arrivals occur before a service occurs, is the same as the probability that a geometrically distributed random variable with success probability  $\mu/(\lambda + \mu) = 1 - p$  takes the value  $j$ .

**s.5.6.5.** Take  $\alpha = \lambda/(\lambda + \mu)$  so that  $f(j) = (1 - \alpha)\alpha^j$ .

$$\begin{aligned} G(j) &= \sum_{k=j+1}^\infty f(k) = (1 - \alpha) \sum_{k=j+1}^\infty \alpha^k \\ &= (1 - \alpha) \sum_{k=0}^\infty \alpha^{k+j+1}, \text{ by change of variable} \\ &= (1 - \alpha) \sum_{k=0}^\infty \alpha^k \alpha^{j+1} = (1 - \alpha) \alpha^{j+1} \sum_{k=0}^\infty \alpha^k \\ &= (1 - \alpha) \alpha^{j+1} \frac{1}{1 - \alpha} = \alpha^{j+1}. \end{aligned}$$

**s.5.6.6.** A simple numerical method is as follows. Make a grid of size  $dx$ , for some small number  $dx$ , e.g.  $dx = 1/100$ , and write  $f_i = P(S \in (i dx, (i + 1) dx]) = F((i + 1) dx) - F(i dx)$ . Then

$$P(Y_k = j) = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!} dF(x) \approx \sum_{i=1}^\infty e^{-\lambda i dx} \frac{(\lambda i dx)^j}{j!} f_i dx.$$

Let's try a numerical experiment.

```
>>> import numpy as np

>>> labda = 3
>>> mu = 4
>>> j = 5
>>> dx = 1 / 100

>>> def F(x):
...     return 1 - np.exp(-mu * x)
...

>>> def f(x):
...     return F(x + dx) - F(x)
...

>>> def term(i):
...     res = np.exp(-labda * i * dx)
...     res *= (labda * i * dx)**j / np.math.factorial(j)
...     res *= f(i*dx) * dx
...     return res
...

>>> print(sum(term(i) for i in range(50)))
1.1158835007226524e-05
>>> print(sum(term(i) for i in range(500)))
8.098807710055904e-05
>>> print(sum(term(i) for i in range(5000)))
8.098807712728956e-05
```

Since I don't know when to stop the integral I just try a few values; of course, stopping the integration at  $x = 50$  is too small, since  $50 dx = 50/100 = 1/2$ , but I include it for illustrative purposes. Stopping at 500 seems OK, since the results of the last two integrals are nearly the same. This also suggests that  $dx = 1/100$  is sufficiently small. In general, however, one must take care and try various values for  $dx$  and the integration limits.

For more complicated situations it is best to use a numerical library to compute the above integral. These methods have been designed to produce good and reliable results, and, typically, it is very hard to improve these methods. Thus, let's try a real number cruncher.

```
>>> from scipy.integrate import quad
```

```

>>> def g(x):
...     return np.exp(-labda*x) * (labda*x)**j/np.math.factorial(j) * f(x)
...
>>> print(quad(g, 0, np.inf))
(8.098807712760667e-05, 3.4086429822163874e-11)

```

This is the same as our earlier answer.

**s.5.6.7.** Follows directly from [3.4.7].

**s.5.6.8.** By using the definitions and limits developed in Sections 2.3 and 4.3,

$$\begin{aligned}
 \lim_{t \rightarrow \infty} \frac{D(n+1, 0, t)}{t} &= \lim_{t \rightarrow \infty} \frac{D(t)}{t} \frac{D(n+1, t)}{D(t)} \frac{D(n+1, 0, t)}{D(n+1, t)} \\
 &= \delta \delta(n+1) \lim_{t \rightarrow \infty} \frac{D(n+1, 0, t)}{D(n+1, t)} \\
 &= \delta \delta(n+1) P(Y = 0) \\
 &= \delta \delta(n+1) f(0),
 \end{aligned}$$

where the last limit follows from the independence of  $Y_k$  and  $L(D_{k-1})$ .

**s.5.6.9.** When  $L(D_{k-1}) > 0$ , job  $k$  is already in the system when job  $k-1$  finishes its service and leaves. Thus, at the departure time  $D_{k-1}$  of job  $k-1$ , the service of job  $k$  can start right away at  $D_{k-1}$ . Then,  $D_k = D_{k-1} + S_k$ .

When job  $k-1$  leaves an empty system behind,  $D_k = A_k + S_k$ , since job  $k$  sees an empty system, hence its service can start right away after its arrival time at time  $A_k$ . Since the arrival process is Poisson by assumption, the time to the next arrival after  $D_{k-1}$  is exponentially distributed with rate  $\lambda$ . (Recall the memoryless property of the inter-arrival times.) Thus,  $A_k - D_{k-1}$  has the same distribution as  $X_k$ , so that  $P(D_k - D_{k-1} \leq x) = P(X_k + S_k \leq x)$ .

**s.5.6.10.** With the definition of the hint,

$$\begin{aligned}
 \lim_{t \rightarrow \infty} \frac{D(0, n, t)}{t} &= \lim_{t \rightarrow \infty} \frac{D(t)}{t} \frac{D(0, t)}{D(t)} \frac{D(0, n, t)}{D(0, t)} \\
 &= \delta \delta(0) \lim_{t \rightarrow \infty} \frac{D(0, n, t)}{D(0, t)} \\
 &= \delta \delta(0) P(Y > n) \\
 &= \delta \delta(0) G(n).
 \end{aligned}$$

$$\begin{aligned}
 \lim_{t \rightarrow \infty} \frac{D(m, n, t)}{t} &= \lim_{t \rightarrow \infty} \frac{D(t)}{t} \frac{D(m, t)}{D(t)} \frac{D(m, n, t)}{D(m, t)} \\
 &= \delta \delta(m) \lim_{t \rightarrow \infty} \frac{D(m, n, t)}{D(m, t)} \\
 &= \delta \delta(m) P(Y > n - m + 1) \\
 &= \delta \delta(m) G(n - m + 1).
 \end{aligned}$$



**s.5.6.11.** Use the shorthands of the hint. We work in steps. First we check whether  $\pi(0) = 1 - \rho$  of the  $M/M/1$  queue satisfies (5.6.6), then we check  $\pi(1)$ , and finally we check the general case with  $n \geq 1$ .

Take  $n = 0$ , so that  $f(0) = 1 - \alpha$ . Hence, the left-hand side becomes  $f(0)\pi(1) = (1 - \alpha)\pi(1)$ . The right-hand side is  $\pi(0)G(0) = \pi(0)\alpha$ . Thus,  $\pi(1) = \pi(0)\alpha/(1 - \alpha) = \rho\pi(0)$ . And indeed, for the  $M/M/1$  queue we know that this is true.

Now we check it for  $n = 1$ ,

$$\begin{aligned}(1 - \alpha)\pi(2) &= \pi(0)G(1) + \pi(1)G(1) = \pi(0)G(1)(1 + \rho) \\ &= \pi(0)\alpha^2(1 + \rho) = \pi(0)\alpha\alpha(1 + \rho) = \pi(0)\alpha\rho.\end{aligned}$$

Dividing by  $1 - \alpha$ , we get

$$\pi(2) = \pi(0)\rho^2.$$

Finally, we fill in  $\pi(n) = \rho^n\pi(0)$  of the  $M/M/1$  queue for  $n \geq 1$ . For ease, we divide both sides by  $\pi(0)$  first. We are left with checking that

$$\begin{aligned}(1 - \alpha)\rho^{n+1} &= \alpha^{n+1} + \sum_{m=1}^n \rho^m \alpha^{n-m+2} \\ &= \alpha^{n+1} + \alpha^{n+2} \sum_{m=1}^n (\rho/\alpha)^m \\ &= \alpha^{n+1} + \alpha^{n+1} \rho \sum_{m=0}^{n-1} (\rho/\alpha)^m \\ &= \alpha^{n+1} + \alpha^{n+1} \rho \frac{1 - (\rho/\alpha)^n}{1 - \rho/\alpha} \\ &= \alpha^{n+1} - \alpha^{n+1}(1 - (\rho/\alpha)^n), \quad \text{as } \rho/\alpha = 1 + \rho, \\ &= \alpha^{n+1}(\rho/\alpha)^n = \alpha\rho^n.\end{aligned}$$

Since  $\rho = \alpha/(1 - \alpha)$  we see that the left- and right-hand sides are the same.

Thus we get that, by using PASTA,  $p(n) = \pi(n) = \rho^n\pi(0) = \rho^n(1 - \rho)$ ; a result we obtained earlier for the  $M/M/1$  queue.



## QUEUEING CONTROL AND OPEN NETWORKS

---

In the queueing systems we analyzed up to now, the server is always present to serve jobs in the system. However, this condition is not always satisfied. As an example, consider a queueing system in which there is a cost associated with switching on and off the server. For instance, in some cases the server has to be set-up for operation; in other cases, the operator of a machine has to move from one place in the factory to another. To reduce the cost, the so-called  $N$ -policy of [1.2.9] can be used. Recall that this policy works as follows. As soon as the system becomes empty (and the server idle), we switch off the server. Then we wait until  $N$  or more jobs have arrived, and then we switch on the server. The server processes jobs until the system is empty again, switches off, and remains idle until a sufficient number of new jobs have arrived, and so on. Thus, we use an  $N$ -policy to *control* the queueing system, in particular the server, and the task is to find a switching threshold  $N$  that minimizes the long-run average cost.

Observe that under such policies the server also has longer busy and idle times. In fact, this sometimes seems to be the policy at dentists or hospitals: do something else until the waiting room is quite full, and then start serving patients. Like this, in the example of a GP, the server (GP) does not have to wait for short times for patients that might be late, but instead can collect idle times into one long stretch, and do something useful instead.

In this chapter we first study the  $M/M/1$  queue and  $M/G/1$  under a  $N$ -policy. As a second topic we study open networks of stations of  $M/M/c$  stations and jobs arrive as a Poisson process. As we will see, the analysis of the  $N$ -policy requires to solve an equation of the type  $v = c + Pv$ , where  $v$  and  $c$  are vectors and  $P$  a (stochastic) matrix, while for the network we need to solve an equation of the type  $\lambda = \gamma + \lambda P$ , where  $\lambda$  and  $\gamma$  are vectors and  $P$  is again a (stochastic) matrix. As these equations are (nearly) the same, we therefore concentrate in Section 6.4 on the solution. The analysis in this chapter allows us to illustrate many tools and results of the previous chapters such as the renewal-reward theorem. In a sense, everything comes together here.

We finally point out that the techniques developed in this chapter extend (way) beyond just queueing theory; they are worth memorizing. The concepts we introduce here can for instance be generalized to (optimal) stopping problems, which find many applications beyond queueing, such in finance, inventory theory, decision theory, and so on. As another set of extensions, it is possible to make the matrix  $P$  and the vector  $c$  depend on an action one can take in certain states. This idea underlies Markov decision theory, which in turn provides the theoretical basis of a number of machine learning tools such as  $Q$  learning, reinforcement learning, and so on. Thus, while this chapter closes our journey on the study of queueing system, it is a first step toward a much longer journey on the diverse applications of the probability theory.

### 6.1 $N$ -POLICIES FOR THE $M/M/1$ QUEUE

Let us consider the  $M/M/1$  queue in which the server switches off as soon as it becomes idle and it costs  $K$  to switch on. Supposing that each job charges  $h$  Euros per unit time while in the system, it makes sense to build up a queue of jobs while the server is idle, and after some

time switch on the server to process jobs until the system is empty again. In particular, we analyze the influence of the  $N$ -policy on the long-run average cost. For this we make a cost model in several steps and at the end we discuss how to minimize the cost as a function of the threshold  $N$ . In passing, we obtain a third way to compute the time-average number  $E[L]$  of jobs in the system; the first resulted from the analysis of the  $M/M/1$  queue in Section 5.1, the second from Little's law, cf. [4.4.8]

### Theory and Exercises

Suppose the server is on, and there are  $q$  jobs in the system. Let us write  $T(q)$  for the expected time to clear the system. Now one of two events happens first. Either a new job enters the system, or the job in service leaves. It follows from [1.3.16] that  $\alpha = \lambda/(\lambda + \mu)$  is the probability the first event occurs and  $\beta = \mu/(\lambda + \mu)$  is the probability the second occurs. Moreover, from [1.3.15] we see that the expected time to either an arrival or a departure, whichever is first, is  $1/(\mu + \lambda)$ . Therefore,  $T(q)$  must satisfy the following recursion:

$$T(q) = \alpha T(q + 1) + \beta T(q - 1) + \frac{1}{\lambda + \mu}. \quad (6.1.1)$$

The reader should note that this type of recursion is a difference equation. Moreover, the ideas behind its derivation are very similar to the ideas used in dynamic programming.

**6.1.1.** *Provide an intuitive explanation for (6.1.1) (where is the memoryless property used?), and show that  $T(q) = q/(\mu - \lambda)$  solves (6.1.1).*

With the same line of reasoning we can compute the expected cost  $V(q)$  to clear the system. Noting that the queueing cost is  $hq$  per unit time when there are  $q$  jobs in the system,  $V(q)$  must satisfy the relation

$$V(q) = \alpha V(q + 1) + \beta V(q - 1) + h \frac{q}{\lambda + \mu}. \quad (6.1.2)$$

To solve this, observe first that in (6.1.1), which we use to compute  $T(q)$ , the last term is a constant, and that  $T(q)$  is a linear function in  $q$ . In the recursion for  $V(q)$  we see that the last term is linear in  $q$ . So let us guess that  $V(q)$  is quadratic in  $q$ , i.e.,  $V(q) = aq^2 + bq + c$ . Thus, we substitute this into the above expression and then try to solve for  $a, b$  and  $c$ .<sup>1</sup> As  $V(0) = 0$ , it follows already that  $c = 0$ . Thus, it remains to find  $a$  and  $b$ .

**6.1.2.** *Use the ideas of [6.1.1] to show that*

$$V(q) = aq^2 + bq = \frac{h}{2} \frac{1}{\mu - \lambda} q^2 + \frac{h}{2} \frac{\lambda + \mu}{(\mu - \lambda)^2} q.$$

Interestingly, the above turns out to be immediately useful. Suppose we switch on the server when  $N = 1$ , that is, directly at the arrival of the first job after the system became empty. Write  $C(1)$  for the expected duration of a cycle that starts when the system becomes idle and stops when the system becomes idle again (and after at least one job has arrived). In other words, the cycle consists of an idle and a busy period.

<sup>1</sup> The reader might wonder about the uniqueness of the solution. Noting that  $V(q + 1)$  follows directly from  $V(q)$  and  $V(q - 1)$ , there can be just one solution when the boundary conditions are given.

**6.1.3.** Explain that  $C(1) = 1/\lambda + T(1)$  for the M/M/1 queue.

**6.1.4.** Use the renewal-reward theorem to explain the relation

$$\frac{V(1)}{C(1)} = \frac{V(1)}{1/\lambda + T(1)} = h E[L],$$

where  $E[L]$  is the expected number of jobs in an M/M/1 queue and given by (5.1.2). Then use the above expressions for  $V(1)$  and  $C(1)$  to verify this.

With this result, we have found yet another way to compute the expected number of jobs in the system.

It remains to generalize to a general threshold  $N$ ; just above we already covered the case with  $N = 1$ . As we already have expressions for the cost and time for the time the server is on, we only have to consider the cost and time while the server is off. Note that right after the server switches off, we need  $N$  independent inter-arrival times to reach level  $N$ . By [1.3.11], the expected time to switch on must be equal to  $N/\lambda$ .

For the cost while building up the queue, we use again a recursive procedure. Write  $W(q)$  for the accumulated queueing cost from the moment the server becomes idle up to the arrival time of the  $q$ th job (the job that sees  $q - 1$  jobs in the system). Then, by the next exercise,

$$W(q) = W(q - 1) + h \frac{q - 1}{\lambda} = h \frac{q(q - 1)}{2\lambda}. \quad (6.1.3)$$

**6.1.5.** Explain the above recursion, and derive the right-hand side.

It remains to assemble all results. Let us assume that the switching cost is  $K$ . Then, by the renewal-reward theorem, the time-average cost of the  $N$ -policy is equal to

$$\frac{W(N) + K + V(N)}{C(N)},$$

where  $C(N) = N/\lambda + T(N)$ .

Finding the optimal  $N$  is easy (from a practical point). Observe that  $V(N)$  and  $W(N)$  are quadratic in  $N$ , while  $C(N)$  is linear in  $N$ . Hence, the average cost is a convex function of  $N$ , and locating the minimizer of a one-dimensional convex function is simple.

### Hints

**h.6.1.1.** We just guess the form  $T(q) = aq + b$ , i.e., linear in  $q$ . It is clear that  $T(0) = 0$ : if the system is empty, the time to clear is zero. Thus,  $b = 0$ , and it remains to solve for  $a$ .

**h.6.1.2.** Fill in our guess  $V(q) = aq^2 + bq$  in the recursion (6.1.2). Then, match the coefficients of  $q^2$ ,  $q$  and the constants, and use that  $\alpha + \beta = 1$ .

**h.6.1.3.** What is the expected time to a job just after the server becomes idle? (Use the memory-less property.)

**h.6.1.4.** What is the cost of one cycle? What is the duration of one cycle?

**h.6.1.5.** Use (0.2.1d).

*Solutions*

**s.6.1.1.** Jobs come in at rate  $\lambda$  and are served at rate  $\mu$ . Therefore the system drains at rate  $\mu - \lambda$ . When there are  $q$  jobs in the system, it must take  $q/(\mu - \lambda)$  units of time to serve these jobs.

We substitute the guess  $T(q) = aq$  in (6.1.1), and solve for  $a$ . Specifically,

$$aq = \alpha(aq + a) + \beta(aq - a) + 1/(\lambda + \mu).$$

Noticing that  $\alpha + \beta = 1$ , this reduces to  $0 = a(\alpha - \beta) + 1/(\lambda + \mu)$ . The rest is trivial.

The reasoning behind (6.1.1) is crucially dependent on the memoryless property. Independent of when we ‘look at the system’, the expected time to the next arrival is  $1/\lambda$  and to the next departure it is  $1/\mu$ , and the expected time to either of these events is  $1/(\lambda + \mu)$ .

**s.6.1.2.** Filling our guess  $V(q) = aq^2 + bq$  in the recursion (6.1.2) gives

$$aq^2 + bq = \alpha a(q^2 + 2q + 1) + \alpha b(q + 1) + \beta a(q^2 - 2q + 1) + \beta b(q - 1) + hq/(\lambda + \mu).$$

Matching the coefficients of  $q^2$ ,  $q$  and the constants, and using that  $\alpha + \beta = 1$ , we obtain

$$\begin{aligned} a &= \alpha a + \beta a & \implies a &= a, \\ b &= \alpha a 2 + \alpha b - \beta a 2 + \beta b + h/(\lambda + \mu), & \implies a &= \frac{h}{2} \frac{1}{\mu - \lambda}, \\ 0 &= \alpha a + \alpha b + \beta a - \beta b, & \implies b &= \frac{a}{\beta - \alpha}. \end{aligned}$$

By combining the last two equations we get the answer.

**s.6.1.3.** After clearing the system, we have to wait for a new job. The expected time for this job arrival is  $1/\lambda$ . (How do we use the memoryless property here?) Once the job arrived, the busy period starts with this one job, and the expected busy time is  $T(1)$ .

**s.6.1.4.** First the explanation. On the one hand, the cost of the jobs in the system during one cycle must be  $V(1)$ . The duration of one cycle is  $C(1)$ . By the renewal-reward theorem, the time-average cost must then be  $V(1)/C(1)$ . On the other hand, if the time-average number of jobs in the system is  $E[L]$ , and each job pays  $h$  per unit time, the time-average cost must be  $h E[L]$ .

Now the check; it’s plain algebra:

$$\begin{aligned} \frac{V(1)}{1/\lambda + T(1)} &= \frac{a + b}{1/\lambda + 1/(\mu - \lambda)} \\ &= \left( \frac{h}{2} \frac{1}{\mu - \lambda} + \frac{h}{2} \frac{\lambda + \mu}{(\mu - \lambda)^2} \right) \frac{\lambda(\mu - \lambda)}{\mu} \\ &= \frac{h}{2} \rho \left( 1 + \frac{\lambda + \mu}{\mu - \lambda} \right) = \frac{h}{2} \rho \frac{2\mu}{\mu - \lambda} \\ &= h \frac{\rho}{1 - \rho} = h E[L]. \end{aligned}$$

**s.6.1.5.** The cost up to the  $q$ th job is the cost up to the arrival of job  $q - 1$  plus the cost while there are  $q - 1$  jobs in the system. The time between the arrival of job  $q - 1$  and  $q$  is  $1/\lambda$ .

It is clear that  $W(0) = 0$ . Therefore,  $W(q) = h \sum_{i=1}^q (i - 1) = hq(q - 1)/2\lambda$ .

6.2 N-POLICIES FOR THE  $M/G/1$  QUEUE

Interestingly, we can extend the analysis of Section 5.6 and Section 6.1 to compute the average cost of the  $M/G/1$  queue under an  $N$ -policy. Thus, rather than exponentially distributed service times, we now consider general service times. For the rest, the model and the problem is the same as in Section 6.1.

*Theory and Exercises*

Similar to Section 5.6, we consider the  $M/G/1$  queueing process at departure moments. It is easy to find an expression for the time to clear the queue right after the departure of a job that leaves  $q$  jobs behind. Analogous to (6.1.1) and using the definition of  $f(k) = P(Y = k)$  as the probability that  $Y = k$  jobs arrive during an arbitrary service time, we see that  $T(q)$  must satisfy the relation

$$T(q) = \sum_{k=0}^{\infty} f(k)T(q+k-1) + E[S].$$

To see this, observe that when  $k = 0$  the next level is  $q - 1$  so that it takes  $T(q - 1)$  time to hit level 0. When  $k = 1$ , the level returns to  $q$  after the departure of the job currently in service; consequently, we have not made any progress, hence we still need  $T(q)$  units of time, and so on. Similar to the reasoning in Section 6.1, we guess that  $T(q) = aq + b$ . Since  $T(0) = 0$ , we already have  $b = 0$ .

**6.2.1.** *Substitute  $T(q) = aq$  in the above recursion for  $T(q)$ , and solve for  $a$  to obtain*

$$T(q) = \frac{E[S]}{1 - E[Y]} q = \frac{E[S]}{1 - \lambda E[S]} q.$$

Let  $V(q)$  be the cost to clear the system right after a departure that leaves  $q$  jobs behind. Thus,  $V(0) = 0$ . Again, analogous Section 5.6 and the derivation above for  $T(q)$ , we see that  $V(q)$  must satisfy the relation

$$V(q) = \sum_{k=0}^{\infty} f(k)V(q+k-1) + H(q), \quad (6.2.1)$$

where  $H(q)$  is the queueing, or holding, cost, which we will determine next.

The queueing cost  $H(q)$  consists of two components. The first is the cost to keep  $q$  jobs in the system while there is a job in service. Clearly, the expected cost of this is  $hq E[S]$ . The second component is the cost of new jobs that arrive during the service. While this is slightly harder to determine, we can combine the ideas underlying the derivation in (6.1.3) and [1.1.6] and [1.3.16]. Specifically, assume it is given that the service time is  $s$ . Then the expected total queueing cost  $U(s)$  of the new arrivals must satisfy for some  $0 < \delta \ll 1$

$$U(s) = U(s - \delta) + (1 - \lambda\delta) \cdot 0 + \lambda\delta hs + o(\delta).$$

This follows because during the interval  $[0, \delta]$  a job arrives with probability  $\lambda\delta$  and stays  $s$  time units in the system, while with probability  $1 - \lambda\delta$  no job arrives, and the cost of this is 0. By the next problem,

$$U(s) = \lambda h \frac{s^2}{2}.$$

**6.2.2.** *Assuming that  $U(\cdot)$  is differentiable, we can use that  $(U(s) - U(s - \delta))/\delta \approx U'(s)$ . Use this in the above to arrive at the differential equation  $U'(s) = \lambda hs$  that  $U$  must satisfy with condition  $U(0) = 0$  (why is this so?). Solve this DE to conclude that  $U(s)$  is as given above.*

Since  $U(s)$  is the expected total cost given that the service time is  $s$ , it follows from [5.6.7] that  $E[U(S)] = \lambda h E[S^2]/2$  is the expected queueing cost of the new jobs that arrive during the service. By combining the first and second component of  $H(q)$ , we obtain

$$H(q) = hq E[S] + \frac{1}{2} \lambda h E[S^2].$$

To solve (6.2.1), note that as in (6.1.2), the cost  $H(q)$  has a term linear in  $q$  and a constant term. Thus, once again, we guess that  $V(q)$  is a quadratic function in  $q$ . Then we substitute this form into (6.2.1), assemble terms with the same power in  $q$ , and try to solve for the coefficients in  $aq^2 + bq + c$ . If we succeed, the solution we found must be correct, as a difference equation of the type (6.2.1) can have just one solution once the boundary conditions are given. Again,  $V(q) = 0 \implies c = 0$ . In fact, observe that we have all elements to compute the costs; except for (quite a bit of) algebra, we are done!

**6.2.3.** Assemble all terms of  $V(q) = \sum_k V(q+k-1) + H(q)$  by their power in  $q$  to obtain:

$$\begin{aligned} aq^2 &= aq^2, \\ bq &= 2aq E[Y] - 2aq + bq + hq E[S], \\ 0 &= a E[Y^2] - 2a E[Y] + a + b E[Y] - b + \frac{1}{2} \lambda h E[S^2]. \end{aligned}$$

**6.2.4.** Simplify the above to obtain

$$\begin{aligned} a &= \frac{h E[S]}{2(1 - E[Y])}, \\ b(1 - E[Y]) &= a(E[Y^2] - 2E[Y] + 1) + \frac{1}{2} \lambda h E[S^2]. \end{aligned}$$

**6.2.5.** Show that  $E[Y^2] = \lambda^2 E[S^2] + \lambda E[S]$ .

**6.2.6.** Show that

$$a = \frac{h E[S]}{2(1 - \rho)}, \quad b = \frac{h E[S]}{2(1 - \rho)^2} (1 + \rho C_s^2).$$

(In a sense, this is trivial, as it is just algebra, but it is hard to get the details right.)

**6.2.7.** We need to check our work. For this reason, show that the results of [6.2.6] reduce to those of [6.1.2] for the  $M/M/1$  queue.

**6.2.8.** As in Section 6.1, show that we can obtain the Pollaczek-Khinchine equation (5.4.1) with the results we have obtained up to now.

For the  $M/G/1$  queue it is evident that the expected queueing cost while the server is idle is also given by (6.1.3). With this, and noting that there is a cost  $K$  to switch on the server, it follows from [6.2.9] that the long-run time-average average costs are equal to

$$\frac{V(N) + K + W(N)}{C(N)} = \frac{h}{2} \frac{\rho^2}{1 - \rho} (1 + C_s^2) + h\rho + h \frac{N-1}{2} + K \frac{\lambda(1 - \rho)}{N}. \quad (6.2.2)$$

Note that, if  $N = 1$  and  $K = 0$ , this reduces to  $h E[L]$ , as it should by our work above. Finally, minimizing over  $N$  gives that

$$N^* \approx \sqrt{\frac{2\lambda(1 - \rho)K}{h}}.$$



**Remark 6.2.1.** The expression for  $N^*$  is a famous result in inventory theory: it is the optimal order size for a machine that produces items with holding cost  $h$  per item per unit time, a cost  $K$  to switch on the machine, demand arrives at rate  $\lambda$ , and the machine produces at rate  $\mu = 1/\mathbb{E}[S]$ .  $N^*$  is known as the Economic Production Quantity (EPQ). Taking  $\mu \rightarrow \infty$ , it reduces to the Economic Order Quantity (EOQ).

**6.2.9.** Derive (6.2.2).

**6.2.10.** Why does the load  $\rho$  not depend on  $N$ ?

*Hints*

**h.6.2.1.** Use that  $\sum_{k=0}^{\infty} k f(k) = \mathbb{E}[Y]$ . Then use [5.6.7]. Realize also that  $\sum_k f(k) = 1$ .

**h.6.2.3.** Just expand the left- and right-hand sides, and equate the terms with the same power in  $q$ .

**h.6.2.8.** What is the cycle time  $C(1)$  when the service starts with one job in the system?

*Solutions*

**s.6.2.1.**

$$aq = \sum_{k=0}^{\infty} a(q+k-1)f(k) + \mathbb{E}[S] \implies aq = aq + a\mathbb{E}[Y] - a + \mathbb{E}[S].$$

The terms linear in  $q$  cancel. And thus,  $0 = a\mathbb{E}[Y] - a + \mathbb{E}[S]$ . The result follows right away.

**s.6.2.2.** Is it plain algebra:

$$\begin{aligned} U(s) &= U(s-\delta) + (1-\lambda\delta) \cdot 0 + \lambda\delta h s + o(\delta) && \implies \\ (U(s) - U(s-\delta))/\delta &= \lambda h s + o(\delta) && \implies \\ U'(s) &= \lambda h s, \quad \text{as } \delta \rightarrow 0. \end{aligned}$$

Integrating both sides with respect to  $s$  gives  $U(s) - U(0) = \lambda h s^2/2$ . As  $U(0) = 0$ , we obtain the final result.

**s.6.2.3.** Just expand

$$\sum_{k=0}^{\infty} f(k)[a(q+k-1)^2 + b(q+k-1)] + H(q),$$

and use that  $\sum_k f(k)k = \mathbb{E}[Y]$  and  $\sum_k f(k)k^2 = \mathbb{E}[Y^2]$ .

**s.6.2.4.** The first equation in the previous equation is superfluous. In the second we can cancel  $bq$  at both sides. Simplifying gives us the expression for  $a$ . Similar for the third equation.

**s.6.2.5.** This follows directly from [5.6.7].

**s.6.2.6.** We already derived  $a$ , except that we replaced  $1 - E[Y] = 1 - \lambda E[S] = 1 - \rho$ . For  $b$ , and using the expressions for  $E[Y]$  and  $E[Y^2]$ ,

$$\begin{aligned}
 b(1 - E[Y]) &= a(E[Y^2] - 2E[Y] + 1) + \frac{1}{2}h\lambda E[S^2] \\
 &= \frac{hE[S]}{2(1 - E[Y])}(E[Y^2] - 2E[Y] + 1) + \frac{1}{2}h\lambda E[S^2] \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} \left( \lambda^2 E[S^2] + \lambda E[S] - 2\lambda E[S] + 1 + \lambda E[S^2] \frac{1 - \lambda E[S]}{E[S]} \right) \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} \left( \lambda^2 E[S^2] - \lambda E[S] + 1 + \frac{\lambda E[S^2]}{E[S]} - \lambda^2 E[S^2] \right) \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} \left( 1 + \frac{\lambda E[S^2]}{E[S]} - \lambda E[S] \right) \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} \left( 1 + \frac{\lambda E[S^2]}{E[S]} - \lambda \frac{(E[S])^2}{E[S]} \right) \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} \left( 1 + \lambda \frac{V[S]}{E[S]} \right) \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} \left( 1 + \lambda \frac{V[S]}{(E[S])^2} E[S] \right) \\
 &= \frac{hE[S]}{2(1 - \lambda E[S])} (1 + \rho C_s^2).
 \end{aligned}$$

Divide now both sides by  $1 - E[Y]$ .

**s.6.2.7.** For  $a$ , multiply the numerator and denominator by  $\mu = 1/E[S]$ . For  $b$ , multiply by  $\mu^2 = 1/(E[S])^2$ , use that  $C_s^1 = 1$  because the service times are exponentially distributed, and note that

$$\frac{1 + \rho}{1 - \rho} = \frac{\mu + \lambda}{\mu - \lambda}.$$

**s.6.2.8.** Right after the arrival of the first job that sees an empty system, the service of this job starts. Thus,  $C(1) = 1/\lambda + T(1)$  is the expected duration of a cycle, and  $V(1)$  the expected queueing cost of once cycle. The rest is algebra:

$$\begin{aligned}
 \frac{V(1)}{1/\lambda + T(1)} &= \frac{a + b}{1/\lambda + E[S]/(1 - \lambda E[S])} \\
 &= \frac{hE[S]}{2} \frac{1 + (1 + \rho C_s^2)/(1 - \rho)}{1 - \rho} \frac{1/\lambda + E[S]/(1 - \rho)}{1/\lambda + E[S]/(1 - \rho)} \\
 &= \frac{hE[S]}{2} \frac{1 - \rho + 1 + \rho C_s^2}{1 - \rho} \frac{(1 - \rho)/\lambda + E[S]}{(1 - \rho)/\lambda + E[S]} \\
 &= \frac{hE[S]}{2} \frac{1 - \rho + 1 + \rho C_s^2}{1 - \rho} \frac{1/\lambda}{1/\lambda} \\
 &= \frac{h}{2} \frac{\rho}{1 - \rho} (1 - \rho + 1 + \rho C_s^2) \\
 &= \frac{h}{2} \frac{\rho}{1 - \rho} (2 - 2\rho + \rho + \rho C_s^2) \\
 &= h\rho + \frac{h}{2} \frac{\rho^2}{1 - \rho} (1 + C_s^2).
 \end{aligned}$$

It checks!

**s.6.2.9.** Note first that  $C(N) = N(1/\lambda + E[S]/(1-\rho)) = N/(\lambda(1-\rho))$ . Then,

$$\begin{aligned}
 \frac{V(N)+K+W(N)}{C(N)} &= (aN^2 + bN + K + hN(N-1)/2\lambda) \frac{\lambda(1-\rho)}{N} \\
 &= \frac{h}{2}\rho N + \frac{h}{2} \frac{\rho}{1-\rho} (1 + \rho C_s^2) + \frac{h}{2}(N-1)(1-\rho) + K \frac{\lambda(1-\rho)}{N} \\
 &= \frac{h}{2} \frac{\rho}{1-\rho} (1 + \rho C_s^2) + \frac{h}{2}(N-1+\rho) + K \frac{\lambda(1-\rho)}{N} \\
 &= \frac{h}{2} \frac{\rho}{1-\rho} (\rho + \rho C_s^2 + 1 - \rho) + \frac{h}{2}(N-1+\rho) + K \frac{\lambda(1-\rho)}{N} \\
 &= \frac{h}{2} \frac{\rho^2}{1-\rho} (1 + C_s^2) + \frac{h}{2}\rho + \frac{h}{2}(N-1+\rho) + K \frac{\lambda(1-\rho)}{N} \\
 &= \frac{h}{2} \frac{\rho^2}{1-\rho} (1 + C_s^2) + h\rho + h \frac{N-1}{2} + K \frac{\lambda(1-\rho)}{N}.
 \end{aligned}$$

**s.6.2.10.** The total number  $A(t)$  of job that arrive during  $[0, t]$  does not depend on  $N$ . Since the system does not explode, we must have that  $\delta = \lambda$ , hence the number of jobs served per unit time is, on the long run, what arrives. And thus, by (4.1.1),  $\rho$  does not depend on  $N$ . But, why is this system rate-stable? Here we provide an intuitive explanation. When we switch on the server, the queue ‘drains’ on average at rate  $\mu - \lambda$ . Consequently, no matter how large  $N$ , the expected time of empty the system is finite. And, whenever the system is empty, the stochastic process restarts. As such cycles start over and over again, the queue length can never ‘escape to infinity’.

### 6.3 OPEN SINGLE-CLASS PRODUCT-FORM NETWORKS

Up to now our analysis focused on single-station queueing systems. In many practical situations, however, jobs in a factory or patients in a hospital have to undergo several process steps before they are ‘finished’. One of the simplest models to analyze such situations is to assume that jobs arrive as a Poisson processes, and the service times are exponentially distributed. We will see that it is possible to obtain closed-form expressions for the stationary distribution of jobs at each station. To establish this we will first concentrate on two stations in tandem, and then extend to general networks. We remark that in Section 3.5 we considered tandem networks of  $G/G/c$  queues, but there we could only obtain insight in the expected times, not the full distribution of the number of jobs at each station.

#### *Theory and Exercises*

##### *Tandem Queues*

In [6.3.8] (and the intermediate exercises leading to that result) we (ask you to) prove that the inter-departure times of an  $M/M/1$  queue are also exponentially distributed with rate  $\lambda$ . This is useful because when the first station is an  $M/M/1$  queue, this implies that the arrival process at the second station is also an  $M/M/1$  queue. Stated differently, from the perspective of the second station, it is as if there is no first station. And, if there is a third station with exponentially

distributed service times, this also behaves as an  $M/M/1$  queue, and so on. With this insight, it is easy to see that the average total waiting time in a tandem network of  $M$  stations equals

$$E[W] = \sum_{i=1}^M \frac{E[S_i]}{1 - \rho_i},$$

where  $\rho_i = \lambda E[S_i]$ , and  $E[S_i]/(1 - \rho_i)$  is the expected waiting time at station  $i$ ; note that the arrival rate at each station is  $\lambda$ , due to the topology of the network, i.e., a tandem network.

**6.3.1.** *Why is the output rate of the (stable)  $M/M/1$  queue equal to  $\lambda$  and not  $\mu$ ?*

**6.3.2.** *Why is  $\mu e^{-\mu t}$  not a reasonable density for the inter-departure times? In fact, the simplest guess for the inter-departure density might be  $\lambda e^{-\lambda t}$ ; so this is what we will try to prove below. We will focus on departure moments and use (4.3.7), in particular that departures ‘see’ what arrivals ‘see’, i.e.,  $\delta(n) = \pi(n)$ , and PASTA.*

**6.3.3.** *Show that the probability that a job leaves behind a busy station is  $\rho$ , hence  $1 - \rho$  is the probability to leave an idle server behind.*

**6.3.4.** *If job  $n - 1$ , say, leaves behind an empty system, show that the expected time until the next departure is  $E[D_n - D_{n-1}] = 1/\lambda + 1/\mu$ .*

**6.3.5.** *Show that the density of  $D_n - D_{n-1}$  is*

$$f_{X+S}(t) = \frac{\lambda\mu}{\lambda - \mu}(e^{-\mu t} - e^{-\lambda t})$$

*if the server is idle after  $D_{n-1}$ .*

**6.3.6.** *Show that when the queue is not empty at a departure time, the density of the next inter-departure time is  $f_D(t) = \mu e^{-\mu t}$ .*

**6.3.7.** *Use conditioning on the server being idle or busy at a departure to show that the density of the inter-departure time is  $\lambda e^{-\lambda t}$ .*

**6.3.8.** *Assuming that inter-departure times are independent, prove Burke’s law which states that the departure process of the  $M/M/1$  queue is a Poisson process with rate  $\lambda$ .*

#### Open Networks of $M/M/1$ queues

It is not difficult to extend the above result for tandem networks to general networks of  $M/M/1$  queues. For this, we first need to model such networks more formally. In particular, we assume that the probability that a job moves to station  $j$  after completing its service at station  $i$  is independent of anything else, and is given by the number  $P_{ij} \in [0, 1]$ . (This is called Markov routing.) We assemble all these probabilities in a *routing matrix*  $P$  such that  $P_{ij}$  is the element of  $P$  on the  $i$ th row and  $j$ th column. We require that  $\sum_{j=1}^M P_{ij} \in [0, 1]$  for each row  $i$ .

**6.3.9.** *Why do we require this? What is the interpretation of  $P_{i0} = 1 - \sum_{j=1}^M P_{ij}$ ? What does it mean when  $P_{i0} > 0$ ?*

Consider station  $i$ , say, and assume that jobs arrive as a Poisson process with rate  $\lambda_i$ . Since service times are exponentially distributed, it follows from the previous section that the departure process is also Poisson with rate  $\lambda_i$ . Then, after departure, jobs are sent with

probability  $P_{ij}$  to station  $j$ , independent of anything else. But then we can use [1.1.18] to conclude that the jobs sent to station  $j$  form a Poisson processes with rate  $\lambda_i P_{ij}$ . Now take the perspective of some station  $j$ . Suppose this station receives such thinned Poisson ‘streams’ from all other stations. Then observe that, by [1.1.14], this merged process is also a Poisson process with the combined rate of the individual ‘streams’. Assuming that new jobs arrive at station  $j$  as a Poisson process with rate  $\gamma_j$ , we can merge this process with the departure processes of the other stations to obtain the total arrival process at station  $j$ , and this must again be Poisson and has rate

$$\gamma_j + \sum_{i=1}^M \lambda_i P_{ij}.$$

Finally, since jobs arrive at station  $j$  as a Poisson process, and service times are exponential, the departure process of this station is also Poisson, and so on for all the stations in the network. Thus, it is intuitively clear that we can model this network as a set of  $M/M/1$  queues; below we will give a formal proof of this fact for two stations. Note that we allow for external jobs arriving at any station. Therefore this network is *open*. This differs from so-called *emph* closed networks; in such networks jobs do not enter or leave.

It is evident that, when the network is stable (so that queues do not keep on increasing over time), all jobs that enter the network must eventually leave. This insight leads us to the *traffic (rate) equations*, which state that for all stations  $i$ ,

$$\lambda_i = \gamma_i + \sum_{j=1}^M \lambda_j P_{ji}, \quad i = 1, \dots, M, \quad (6.3.1)$$

where the left-hand side represents the rate at which jobs depart and the right-hand side the rate at which jobs arrive.

Let us for the moment assume that we can solve the traffic equations, in other words, we can find a set of numbers  $\lambda = (\lambda_1, \dots, \lambda_M)$  such that (6.3.1) is satisfied for given  $\gamma = (\gamma_1, \dots, \gamma_M)$  and routing matrix  $P$ , cf. Section 6.4. Then, we can define the load at station  $i$  as  $\rho_i = \lambda_i E[S_i]$ . Clearly, we assume that  $\rho_i < 1$  for all stations  $i$ . Moreover, station  $i$  being an  $M/M/1$  queue, it follows that the sojourn time is  $E[W_i] = E[S_i]/(1 - \rho_i)$ . Write  $|\gamma| = \sum_{i=1}^M \gamma_i$  as the total external arrival rate. Then, using that the average total number of jobs is equal to the sum of the average number of jobs at each station, it must follow that

$$E[L] = \sum_{i=1}^M E[L_i].$$

Then with an application of Little’s law to the network as a whole and to each station individually, we get

$$|\gamma| E[W] = E[L] = \sum_{i=1}^M E[L_i] = \sum_{i=1}^M \lambda_i E[W_i].$$

As a last step, by dividing by  $|\gamma|$ , we can express the average sojourn time in the network in terms of the *visiting ratios*  $\lambda_i/|\gamma|$  as

$$E[W] = \sum_{i=1}^M \frac{\lambda_i}{|\gamma|} E[W_i].$$

**6.3.10.** *Provide an interpretation for the above expression.*

**6.3.11.** We have a two-station single-server open network. Jobs enter the network at the first station with rate  $\gamma$ . A fraction  $\alpha$  returns from station 1 to itself; the rest moves to station 2. At station 2 a fraction  $\beta_2$  returns to station 2 again, a fraction  $\beta_1$  goes to station 1.

First, compute  $\lambda$ , then analyze what happens if  $\alpha \rightarrow 1$  or  $\beta_1 \rightarrow 0$ .

#### Stationary distributions

Above we derived expressions for the average waiting time in a network of  $M/M/1$  queues. In fact, it is possible to obtain the much stronger result that the stationary probability that the system contains  $n = (n_1, n_2, \dots, n_M)$  at stations  $1, \dots, M$  takes the form

$$P(N_1 = n_1, \dots, N_M = n_M) = p(n) = \prod_{i=1}^M p(n_i) = \prod_{i=1}^M (1 - \rho_i) \rho_i^{n_i},$$

where  $p(n_i) = (1 - \rho_i) \rho_i^{n_i}$  is the stationary probability that station  $i$  contains  $n_i$  jobs, compare (5.1.1). In words,  $p(n)$  is equal to the product of the probabilities  $p(n_i)$  of all of the stations  $i = 1, \dots, M$ . But, this implies that  $p(n_i)$  is *independent* of the state of the other stations. For notational ease, we will prove this for the case of two stations in tandem. The general result is known as the fact that *Jackson networks*, i.e., open networks of  $M/M/c$  networks, admit a *product-form solution*. Note that these probabilities are useful to estimate excess probabilities such as  $P(L_1 > n_1, L_2 > n_2)$ .

Write  $p(i, j) = P(N_1 = i, N_2 = j)$  for the state of the two-station network to denote that station 1 contains  $i$  jobs and station 2 contains  $j$  jobs. We have to show that the probabilities  $p(i, j)$  satisfy the balance equations for all  $i, j \geq 0$ . Recall that the balance equations express that, in steady state, the total (probability) rate out of a state must be equal to the (probability) rate into this state, cf., Section 4.2.

**6.3.12.** Provide the balance equation for state  $(0, 0)$  and check that this is satisfied by  $p(0, 0)$ .

**6.3.13.** Provide the balance equations for states  $(i, 0)$  with  $i \geq 1$  and check that these are satisfied by  $p(i, 0)$ .

**6.3.14.** Provide the balance equations for states  $(0, j)$  with  $j \geq 1$  and check that these are satisfied by  $p(0, j)$ .

**6.3.15.** Provide the balance equations for states  $(i, j)$  with  $i, j \geq 1$  and check that these are satisfied by  $p(i, j)$ .

**6.3.16.** Provide the balance equations for states  $(i, j)$  with  $i, j \geq 0$  and check that these are satisfied by  $p(i, j)$  for  $i, j \geq 0$ .

#### Hints

**h.6.3.4.** You might find some inspiration in [5.6.9].

**h.6.3.7.** Conditioning leads to

$$f_D(t) = f_{X+S}(t)P(\text{server is idle}) + f_S(t)P(\text{server is busy}) = (1 - \rho)f_{X+S}(t) + \rho\mu e^{-\mu t}.$$

Now use the above exercises to simplify.

**h.6.3.10.** What is the intuition behind the visit ratios?

**h.6.3.12.** Realize that an arrival is required to leave state  $(0, 0)$ , and a departure at the second queue is necessary to enter state  $(0, 0)$ .

### Solutions

**s.6.3.1.** Jobs arrive at rate  $\lambda$ . For a stable queue,  $\mu > \lambda$ . Moreover, jobs can never leave faster than they arrive. Note that, when there are jobs *present* at station 1, jobs do leave at rate  $\mu_1$ . However, there are not always jobs in service, and during such times, there are no job departures.

**s.6.3.2.** Because jobs do not leave at rate  $\mu$ .

**s.6.3.3.** Observe that  $\rho$  is the fraction of time the server is busy. Then, from PASTA, the fraction of jobs that see a busy server is also  $\rho$ . This fraction of jobs is  $\sum_{n=1}^{\infty} \pi(n)$ . Finally,  $\delta(n) = \pi(n)$ , a fraction  $\rho$  of the departures leaves a busy system behind.

**s.6.3.4.** After job  $n-1$  left, job  $n$  has to arrive, so we need to wait first for this inter-arrival time. Then job  $n$  must be served. This adds up to  $1/\lambda + 1/\mu$ .

In more detail, we first have to wait for an inter-arrival time  $X_n$ . Then, since job  $n$ 's service starts right away, it leaves when  $D_n = D_{n-1} + X_n + S_n$ . Now observe that, due to the memoryless property of the inter-arrival times,  $E[X_n] = E[A_n - D_{n-1}] = 1/\lambda$ . Thus, the expected duration is  $E[X_n + S_n] = 1/\lambda + 1/\mu$ .

**s.6.3.5.** By the previous point, the density of  $D_n - D_{n-1}$  is the same as the density of  $X_n + S_n$ . Since  $\{X_n\}$  and  $\{S_n\}$  are both i.i.d. sequences, the problem becomes to find the density of  $X + S$ . We will use two ways of computing this.

Since  $X \sim \text{Exp}(\lambda)$  and  $S \sim \text{Exp}(\mu)$ , and  $X$  and  $S$  are independent, their joint density is  $f_{X,S}(x,y) = \lambda\mu e^{-\lambda x - \mu y}$ . With this,

$$\begin{aligned}
 P(X + S \leq t) &= \lambda\mu \int_0^\infty \int_0^\infty e^{-\lambda x - \mu y} \mathbb{1}_{x+y \leq t} dx dy \\
 &= \lambda\mu \int_0^t \int_0^{t-x} e^{-\lambda x - \mu y} dy dx \\
 &= \lambda\mu \int_0^t e^{-\lambda x} \int_0^{t-x} e^{-\mu y} dy dx \\
 &= \lambda \int_0^t e^{-\lambda x} (1 - e^{-\mu(t-x)}) dx \\
 &= \lambda \int_0^t e^{-\lambda x} dx - \lambda e^{-\mu t} \int_0^t e^{(\mu-\lambda)x} dx \\
 &= 1 - e^{-\lambda t} - \frac{\lambda}{\mu - \lambda} e^{-\mu t} (e^{(\mu-\lambda)t} - 1) \\
 &= 1 - e^{-\lambda t} - \frac{\lambda}{\mu - \lambda} e^{-\lambda t} + \frac{\lambda}{\mu - \lambda} e^{-\mu t} \\
 &= 1 - \frac{\mu}{\mu - \lambda} e^{-\lambda t} + \frac{\lambda}{\mu - \lambda} e^{-\mu t}.
 \end{aligned}$$

The density  $f_{X+S}(t)$  is the derivative of this expression with respect to  $t$ , hence,

$$\begin{aligned}
 f_{X+S}(t) &= \frac{\lambda\mu}{\mu - \lambda} e^{-\lambda t} - \frac{\mu\lambda}{\mu - \lambda} e^{-\mu t} \\
 &= \frac{\lambda\mu}{\lambda - \mu} (e^{-\mu t} - e^{-\lambda t}).
 \end{aligned}$$

Conditioning is much faster, but requires the concept of conditional density. You can skip the rest if you are not interested.

$$\begin{aligned}
 f_{X+S}(t) &= P(X+S \in dt) \\
 &= \int P(S+x \in dt) P(X \in dx) \\
 &= \int_0^t f_S(t-x) f_X(x) dx \\
 &= \int_0^t \mu e^{-\mu(t-x)} \lambda e^{-\lambda x} dx \\
 &= \lambda \mu e^{-\mu t} \int_0^t e^{x(\mu-\lambda)} dx \\
 &= \frac{\lambda \mu}{\lambda - \mu} (e^{-\mu t} - e^{-\lambda t}).
 \end{aligned}$$

**s.6.3.6.** After the departure, the server can start right away with the job at the head of the queue. The inter-departure time of this job is  $\text{Exp}(\mu)$ .

**s.6.3.7.**

$$\begin{aligned}
 f_D(t) &= (1-\rho)f_{X+S}(t) + \rho \mu e^{-\mu t} \\
 &= (1-\rho) \frac{\mu \lambda}{\lambda - \mu} (e^{-\mu t} - e^{-\lambda t}) + \rho \mu e^{-\mu t} \\
 &= \left(1 - \frac{\lambda}{\mu}\right) \frac{\mu \lambda}{\lambda - \mu} (e^{-\mu t} - e^{-\lambda t}) + \rho \mu e^{-\mu t} \\
 &= \frac{\mu - \lambda}{\mu} \frac{\mu \lambda}{\lambda - \mu} (e^{-\mu t} - e^{-\lambda t}) + \frac{\lambda}{\mu} \mu e^{-\mu t} \\
 &= -\lambda (e^{-\mu t} - e^{-\lambda t}) + \lambda e^{-\mu t} \\
 &= \lambda e^{-\lambda t}.
 \end{aligned}$$

**s.6.3.8.** It follows from [6.3.1] to [6.3.7] that inter-departures times have the same density function  $\lambda e^{-\lambda t}$ . The independence of inter-departure times is assumed. These two facts combined imply that the departure process forms a Poisson process with rate  $\lambda$ .

**s.6.3.9.** A job leaving station  $i$  has to go somewhere, either to another station or perhaps to station  $i$  again, or it leaves the network altogether. When the summation is less than 1, jobs are allowed to leave the network. As  $\sum_{j=1}^M P_{ij}$  is the probability to be sent to some station in the network,  $P_{i0}$  is the probability a job leaves the network. When  $P_{i0} > 0$ , jobs do leave from station  $i$ .

**s.6.3.10.** The visit ratio  $\lambda_i/|\gamma|$  tells how often station  $i$  is visited relative to the total number of arrivals. Thus,  $\lambda_i E[W_i]/|\gamma|$  is the amount of time an ‘average’ job spends at station  $i$  before leaving.

**s.6.3.11.**

$$P = \begin{pmatrix} \alpha & 1-\alpha \\ \beta_1 & \beta_2 \end{pmatrix}.$$



$$(\lambda_1, \lambda_2) = (\gamma, 0) + (\lambda_1, \lambda_2)P.$$

Solving first for  $\lambda_2$  leads to  $\lambda_2 = (1 - \alpha)\lambda_1 + \beta_2\lambda_2$ , so that

$$\lambda_2 = \frac{1 - \alpha}{1 - \beta_2} \lambda_1.$$

Next, using this and that  $\lambda_1 = \alpha\lambda_1 + \beta_1\lambda_2 + \gamma$  gives with a bit of algebra

$$\begin{aligned} \gamma &= \lambda_1(1 - \alpha) - \beta_1\lambda_2 \\ &= \lambda_1 \left( 1 - \alpha - \beta_1 \frac{1 - \alpha}{1 - \beta_2} \right) \\ &= \lambda_1(1 - \alpha) \left( 1 - \frac{\beta_1}{1 - \beta_2} \right) \\ &= \lambda_1(1 - \alpha) \frac{1 - \beta_1 - \beta_2}{1 - \beta_2}. \end{aligned}$$

Hence,

$$\lambda_1 = \frac{\gamma}{1 - \alpha} \frac{1 - \beta_2}{1 - \beta_1 - \beta_2}.$$

Thus,

$$\lambda_2 = \frac{1 - \alpha}{1 - \beta_2} \lambda_1 = \frac{\gamma}{1 - \beta_1 - \beta_2}.$$

We want of course that  $\lambda_1 < \mu_1$  and  $\lambda_2 < \mu_2$ . With the above expressions this leads to conditions on  $\alpha$ ,  $\beta_1$  and  $\beta_2$ . Note that we have three parameters, and two equations; there is not a single condition from which the stability can be guaranteed.

If  $\alpha \uparrow 1$ , the arrival rate at node 1 explodes. If  $\beta_1 = 0$  no jobs are sent from node 2 to node 1.

**s.6.3.12.** Since  $p(i, j) = (1 - \rho_1)(1 - \rho_2)\rho_1^i\rho_2^j$ , we drop the normalization factor in the checks to come.

The rate into state  $(0, 0)$  is  $\mu_2 p(0, 1) = \mu_2 \rho_2$ . The rate out of state  $(0, 0)$  is  $\lambda p(0, 0) = \lambda$ . Since  $\rho_2 = \lambda/\mu_2$ , these two rates are the same.

**s.6.3.13.** We show that the rate out is the rate in.

$$\begin{aligned} \text{rate out} &= (\lambda + \mu_1)p(i, 0) = \mu_1\rho_1^i + \lambda\rho_1^i \\ &= \lambda\rho_1^{i-1} + \mu_2\rho_1^i\rho_2 \\ &= \lambda p(i - 1, 0) + \mu_2 p(i, 1) = \text{rate in}. \end{aligned}$$

**s.6.3.14.** We show that the rate out is the rate in.

$$\begin{aligned} \text{rate out} &= (\lambda + \mu_2)p(0, j) = \mu_2\rho_2^j + \lambda\rho_2^j \\ &= \mu_1\rho_1\rho_2^{j-1} + \mu_2\rho_2^{j+1} \\ &= \mu_1 p(1, j - 1) + \mu_2 p(0, j + 1) = \text{rate in}. \end{aligned}$$

**s.6.3.15.** We show that the rate out is the rate in.

$$\text{rate out} = (\lambda + \mu_1 + \mu_2)p(i, j)$$

$$\begin{aligned}
&= \lambda \rho_1^i \rho_2^j + \mu_1 \rho_1^i \rho_2^j + \mu_2 \rho_1^i \rho_2^j \\
&= \mu_2 \rho_1^i \rho_2^{j+1} + \lambda \rho_1^{i-1} \rho_2^j + \mu_1 \rho_1^{i+1} \rho_2^{j-1} \\
&= \mu_2 p(i, j+1) + \lambda p(i-1, j) + \mu_1 p(i+1, j-1) \\
&= \text{rate in.}
\end{aligned}$$

**s.6.3.16.** See the exercises above.

6.4 ON  $\lambda = \gamma + \lambda P$

Here we study the existence of a solution for the equation  $\lambda = \gamma + \lambda P$ , and its inverse  $V = c + PV$ .

#### Theory and Exercises

From [6.3.9] we know that  $P_{i0} = 1 - \sum_{j=1}^M P_{ij}$  is the probability that a job departing from node  $i$  also leaves the network, in other words, with probability  $P_{i0}$  a job is finished after its service at station  $i$ . Next, consider a station  $k$  with  $P_{ki} > 0$ . Then the probability that a job starting at  $k$ , moving to  $i$  and then leaving the network, must be equal to  $P_{ki}P_{i0}$ . As  $P_{ki} > 0$  and  $P_{i0} > 0$ , the probability that a job leaves the network from node  $k$  in two steps is positive. As a matter of fact,  $P_{k0}^2 = \sum_{j=0}^M P_{kj}P_{j0} \geq P_{ki}P_{i0} > 0$ . More generally, we assume that the (finite) matrix  $P$  is transient, which means that it is possible to leave the network from any station in at most  $M$  steps. In other words, for any station  $j$  there is a sequence of intermediate stations  $j_1, j_2, \dots, j_{M-1}$  such that  $P_{j0}^M \geq P_{jj_1}P_{j_1j_2} \cdots P_{j_{M-1}0} > 0$ .

**6.4.1.** What type of networks lead to the longest chains before jobs can leave the network? What routing matrix  $P$  corresponds to such a network? Show that  $P^M = 0$  for such networks.

The next few exercises provide two different ways to prove that a finite transient matrix  $P^n \rightarrow 0$  geometrically fast ((element wise) as  $n \rightarrow \infty$ ).

**6.4.2.** Use that  $\sum_{j=1}^M P_{ij}^M < 1$  for all  $i = 1, \dots, M$  to prove that  $P^n \rightarrow 0$  geometrically fast in  $n$ .

**6.4.3.** Why does the assumption of [6.4.2] does not apply to an infinite transient matrix  $P$ ?

Before we can provide the second type of proof, do the next exercise.

**6.4.4.** What is the geometric interpretation of an eigenvector and eigenvalue of a matrix  $P$ , say? Specifically, what happens if an eigenvalue has modulus less than 1?

Let us make the simplifying assumption that  $P$  is a diagonalizable matrix with  $M$  different eigenvalues.<sup>2</sup> In this case, there exists an invertible matrix  $V$  with the (left) eigenvectors of  $P$  as its rows and a diagonal matrix  $\Lambda$  with the eigenvalues on its diagonal such that  $VP = \Lambda V$ . Hence, premultiplying with  $V^{-1}$ ,  $P = V^{-1}\Lambda V$ . But then  $P^2 = V^{-1}\Lambda V \cdot V^{-1}\Lambda V = V^{-1}\Lambda^2 V$ , and in general  $P^n = V^{-1}\Lambda^n V$ . Clearly, if each eigenvalue  $\lambda_i$  is such that its modulus  $|\lambda_i| < 1$ , then  $\Lambda^n \rightarrow 0$  geometrically fast, hence  $P^n \rightarrow 0$  geometrically fast.

So, let us prove that all eigenvalues of a finite, transient routing matrix  $P$  have modulus less than 1. For this we use *Gerschgorin's disk theorem*. Define the Gerschgorin disk of the  $i$ th row of the matrix  $P$  as the disk in the complex plan:

$$B_i = \left\{ z \in \mathbb{C}; |z - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\}.$$

<sup>2</sup> The argument below applies just as well matrices reduced to Jordan normal form, but only adds notational clutter.

In words, this is the set of complex numbers that lies within a distance  $\sum_{j \neq i} |a_{ij}|$  of the point  $a_{ii}$ . Next, assume for notational simplicity that for each row  $i$  of  $P$  we have that  $\sum_j a_{ij} < 1$  (otherwise apply the argument to  $P^M$ .) Then this implies for all  $i$  that

$$1 > \sum_{j=1}^M a_{ij} = a_{ii} + \sum_{j \neq i} a_{ij}.$$

Since all elements of  $P$  are non-negative, so that  $|a_{ij}| = a_{ij}$ , it follows that

$$-1 < a_{ii} - \sum_{j \neq i} a_{ij} \leq a_{ii} + \sum_{j \neq i} a_{ij} < 1.$$

With this and using that  $a_{ii}$  is a real number (so that it lies on the real number axis) it follows that the disk  $B_i$  lies strictly within the complex unit circle  $\{z \in \mathbb{C}; |z| \leq 1\}$ . As this applies to any row  $i$ , the union of the disks  $\cup_i B_i$  lies strictly within the complex unit circle. Now Gerschgorin's theorem states that all eigenvalues of the matrix  $P$  must lie in  $\cup_i B_i$ . We conclude that all eigenvalues of  $P$  also lie strictly in the unit circle, hence all eigenvalues have modulus smaller than 1.

With the above results, we can show that the equation  $\lambda = \gamma + \lambda P$  has a unique solution when  $P$  is a transient matrix. For this, define iteratively,

$$\lambda^0 = 0, \quad \lambda^n = \gamma + \lambda^{n-1}P, \quad \text{for } n \geq 1.$$

Then, by substituting  $\lambda^j = \gamma + \lambda^{j-1}P$  a sufficient number of times,

$$\lambda^n = \gamma + \lambda^{n-1}P = \gamma + (\gamma + \lambda^{n-2}P)P = \gamma \sum_{i=0}^{n-1} P^i,$$

where we take  $P^0 = 1$ , i.e., equal to the identity matrix. By the result of the above reasoning, there exists an  $N$  and  $\epsilon > 0$  such that  $P_{ij}^n < (1 - \epsilon)^n$  for all  $n > N$  and  $1 \leq i, j \leq M$ . Therefore, and using (0.2.1d), each element  $i, j$  of the sequence of matrices  $\sum_{k=0}^n P^k$  increases monotonically as  $n \rightarrow \infty$  to a finite limit. Consequently,

$$\lambda = \gamma \sum_{k=0}^{\infty} P^k \tag{6.4.1}$$

is well-defined, finite, and the solution of  $\lambda = \gamma + \lambda P$ .

We can apply the results here also to see why the recursions for  $T$ ,  $V$  and  $W$  in Section 6.1 and Section 6.2 have solutions. For instance, observe that we can write (6.2.1) as

$$V = PV + H,$$

with

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots \\ f(0) & f(1) & f(2) & 0 & \dots \\ 0 & f(0) & f(1) & f(2) & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \quad H = \begin{pmatrix} 0 \\ H(1) \\ H(2) \\ \vdots \end{pmatrix}.$$

It is clear that  $V = PV + H$  is the same equation (in transpose) as  $\lambda = \gamma + \lambda P$ . With the same type of reasoning, we can find the solution for  $V$ , for instance by defining iteratively  $V^0 = 0$ , and

$V^n = PV^{n-1} + H$ , for  $n \geq 0$ . Again,  $P$  is here a non-negative matrix, although it is infinite, which adds a few technical complications. These complications can be solved, and with this body of theory we can analyze all such systems.

We finish our discussion of queueing systems, but there many other interesting extensions to learn. Here are some nice references that are now accessible to you.

- You can find really nice discussion of networks of  $M/M/\infty$ , chemical reactions, population dynamics and Petri nets in [Baez and Biamonte \[2019\]](#), which is freely available on arXiv.
- Simple queueing networks (networks that satisfy so-called local balance) can be modeled as electrical networks. For this, see [Doyle and Laurie Snell \[1984\]](#), which you can download for free from the homepage of Doyle.
- In more general terms, queueing systems or networks are examples of Markov processes. A particularly nice book on these topics is [Norris \[1997\]](#). The material of this chapter can be couched in the theory of martingales and optimal stopping. Besides that this is nice theory, this is widely used in quantitative finance.

### Hints

**h.6.4.1.** Analyze the tandem network.

**h.6.4.4.** Recall that when  $x$  is an eigenvector of  $P$  with eigenvalue  $\lambda$ , then  $Px = \lambda x$ .

### Solutions

**s.6.4.1.** A network of stations is tandem must be the worst, as it is impossible to leave the network in less than  $M$  steps. The routing matrix  $P$  is such that  $P_{i,i+1} = 1$ , and  $P_{ij} = 0$  for  $j \neq i+1$ . As  $P$  is an upper-triangular matrix of dimension  $M \times M$ ,  $P^M = 0$ .

Note that by adding rework, that is, a fraction of jobs is sent upstream into the network, the average length of a routing can be much longer than  $M$ ; however, it is still possible (with small probability perhaps) to leave the network in  $M$  steps.

**s.6.4.2.** By assumption,  $\sum_{j=1}^M P_{ij}^M < 1$  for all  $i = 1, \dots, M$ . Since  $M$  is a finite number, there exists an  $\epsilon > 0$  such that  $\sum_{j=1}^M P_{ij}^M < 1 - \epsilon$  for all rows  $i$ . Writing  $\mathbf{1}$  for the vector  $(1, 1, \dots, 1)$ , this condition means that  $P^M \mathbf{1} < (1 - \epsilon)\mathbf{1}$ . But then, by the linearity of  $P$ ,

$$P^{2M} \mathbf{1} = P^M(P^M \mathbf{1}) < (1 - \epsilon)P^M \mathbf{1} < (1 - \epsilon)^2 \mathbf{1}.$$

In words, this says that each of the row sums become small. As the elements of  $P \geq 0$ , this implies that all elements of  $P^{nM}$  decreases, that is,  $P^{nM} < (1 - \epsilon)^n$  as  $n \rightarrow \infty$ .

**s.6.4.3.** The above argumentation is not necessarily valid for matrices  $P$  that are infinite, since  $\inf\{P_{ik}^M\}$  need not be strictly positive for all  $i, j$ .

**s.6.4.4.** The equation  $Px = \lambda x$  shows that the direction of the vector  $x$  remains the same under  $P$ , only its length changes by  $|\lambda|$ . Thus, if  $|\lambda| < 1$ , the vector  $x$  shrinks under the operation of  $P$ .

## BIBLIOGRAPHY

---

- F. Baccelli and W.A. Massey. A sample path analysis of the  $M/M/1$  queue. *Journal of Applied Probability*, 26(2):418–422, 1988.
- J.C. Baez and J. Biamonte. Quantum techniques for stochastic mechanics, 2019.
- G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley & Sons, 2006.
- M. Capiński and T. Zastawniak. *Probability through Problems*. Springer Verlag, 2nd edition, 2003.
- D.R. Cox, editor. *Renewal Theory*. John Wiley & Sons Inc, New York, 1962.
- P. Doyle and J Laurie Snell. *Random Walks and Electrical Networks*. Mathematical Association of America, 1984.
- M. El-Taha and S. Stidham Jr. *Sample-Path Analysis of Queueing Systems*. Kluwer Academic Publishers, 1998.
- R.W. Hall. *Queueing Methods for Services and Manufacturing*. Prentice Hall, 1991.
- J.R. Norris. *Markov Chains*. Cambridge University Press, 1997.
- H.C. Tijms. *Stochastic Models, An Algorithmic Approach*. J. Wiley & Sons, 1994.
- H.C. Tijms. *A First Course in Stochastic Models*. John Wiley & Sons, Chichester, 2003.
- A.A. Yushkevich and E.B. Dynkin. *Markov Processes: Theorems and Problems*. Plenum Press, 1969.



## NOTATION

---

- $a_k$  = Number of arrivals in the  $k$ th period
- $A(t)$  = Number of arrivals in  $[0, t]$
- $A_k$  = Arrival time of  $k$ th job
- $\tilde{A}_k$  = Start of service of  $k$ th job
- $c_n$  = Service/production capacity in the  $n$ th period
- $d_n$  = Number of departures in the  $n$ th period
- $c$  = Number of servers
- $C_a^2$  = Squared coefficient of variation of the inter-arrival times
- $C_s^2$  = Squared coefficient of variation of the service times
- $D(t)$  = Number of departures in  $[0, t]$
- $D_Q(t)$  = Number of customers/jobs that departed from the queue in  $[0, t]$
- $D_k$  = Departure time of  $k$ th job
- $F$  = Distribution of the service time of a job
- $L(t)$  = Number of customers/jobs in the system at time  $t$
- $Q(t)$  = Number of customers/jobs in queue at time  $t$
- $L_S(t)$  = Number of customers/jobs in service at time  $t$
- $E[L]$  = Long run (time) average of the number of jobs in the system
- $E[L_Q]$  = Long run (time) average of the number of jobs in queue
- $E[L_S]$  = Long run (time) average of the number of jobs in service
- $N(t)$  = Number of arrivals in  $[0, t]$
- $N(s, t)$  = Number of arrivals in  $(s, t]$
- $p(n)$  = Long-run time average that the system contains  $n$  jobs
- $Q_k$  = Queue length as seen by the  $k$ th job, or at the *end* of the  $k$ th period
- $S_k$  = Service time required by the  $k$ th job
- $S(t)$  = Total service time available in  $[0, t]$
- $S$  = Service time of a generic job
- $W_k$  = Sojourn time of  $k$ th job
- $W_{Q,k}$  = Time in the queue of  $k$ th job
- $E[W]$  = Sample average of the sojourn time
- $E[W_Q]$  = Sample average of the time in queue
- $X_k$  = Inter-arrival time between job  $k - 1$  and job  $k$
- $X$  = Generic inter-arrival time between two consecutive jobs
- $\delta$  = Departure rate
- $\lambda$  = Arrival rate

$\mu$  = Service rate

$\pi(n)$  = Stationary probability that an arrival sees  $n$  jobs in the system

$\rho$  = Load on the system



## FORMULA SHEET

---

$$\rho = \lambda \frac{E[S]}{c}$$

$$E[W_Q] = \frac{C_a^2 + C_s^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} E[S]$$

$$\text{Batching: } C_{sB}^2 = \frac{B V[S_0] + V[T]}{(B E[S_0] + E[T])^2}$$

$$\text{Nonpreemptive: } V[S] = V[S_0] + \frac{V[T]}{B} + \frac{B-1}{B^2} (E[T])^2$$

$$\text{Preemptive: } A = \frac{m_f}{m_r + m_f}, C_s^2 = C_0^2 + 2A(1-A) \frac{m_r}{E[S_0]}$$

$$C_{di}^2 = 1 + (1 - \rho_i^2)(C_{ai}^2 - 1) + \frac{\rho_i^2}{\sqrt{c_i}}(C_{si}^2 - 1)$$

$$f_i(n_i) = \frac{(c_i \rho_i)^{n_i}}{n_i! G(i)} \mathbb{1}_{n_i < c_i} + \frac{c_i^{c_i} \rho_i^{n_i}}{c_i! G(i)} \mathbb{1}_{n_i \geq c_i},$$

$$\text{with } G(i) = \sum_{n=0}^{c_i-1} \frac{(c_i \rho_i)^n}{n!} + \frac{(c_i \rho_i)^{c_i}}{c_i!} \frac{1}{1 - \rho_i}$$

$$E[L_i] = \frac{(c_i \rho_i)^{c_i}}{c_i! G(i)} \frac{\rho_i}{(1 - \rho_i)^2} + c_i \rho_i$$



## INDEX

---

- ample server, 48
- arrival process, 32
- arrival rate, 2, 52
- arrival times, 32
- average number of jobs, 57
  
- balance equations, 93
- balking, 116
- binomially distributed, 2
- Burke's law, 176
  
- compound Poisson, 48
- conditional probability, x
  
- departure rate, 53
- departure time of the system, 33
- distribution function, ix
  
- Economic Order Quantity (EPQ), 173
- Economic Production Quantity (EPQ), 173
- effective processing time, 68
- expected sojourn time, 56
- expected waiting time in queue, 56
- exponentially distributed, 26
  
- Gerschgorin's disk theorem, 182
  
- i.i.d., 2
- indicator function, viii
- inter-arrival times, 32
  
- Jackson networks, 178
  
- Kendall's abbreviation, 45
  
- level-crossing equations, 91
- limiting, 57
- load, 54, 87
  
- memoryless, 27
- Merging, 3
- moment-generating function, ix
- multi-server, 47
  
- net processing time, 68
  
- non-preemptive, 73
- normalization constant, 91
- number of jobs in the system, 34
  
- one-step transitions, 99
- open, 177
  
- PASTA, 96
- PK formula, 143
- Poisson arrivals see time averages, 96
- Poisson distributed, 2
- Poisson process, 2
- Pollaczek-Khinchine formula, 143
- probability mass function, viii
- processing rate, 53
- product-form solution, 178
  
- rate-stable, 53
- remaining service time, 144
- renewal reward theorem, 87
  
- SCV, 3
- service rate, 53
- service time, 37
- small  $o$  notation, viii
- sojourn time, 34
- square coefficient of variation, 3
- stationary, 57
- stationary and independent increments, 2
- steady-state, 57
- survivor function, ix
- system capacity, 47
  
- time-average number of jobs, 57
- traffic (rate) equations, 177
  
- utilization, 87
  
- variance, 26
- virtual waiting time process, 34
  
- waiting time in queue, 33
- Wald's equation, 77