Analysis of Queueing Systems with Sample Paths and Simulation

Nicky D. van Foreest

March 6, 2020

Changes:

- Figure 8
- Section 1.7.
- Section 1.3, added python code to a few exercises to show how to implement the recursions.
- Moved part of **??** to **??**.
- Moved old intro of Section 2.4 to 2.1.2.
- Formulas in intro Section 2.6.
- Section 2.9: rewritten
- Section 3.2 restructured

CONTENTS

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 $\pi(4)+\pi(5)+\cdots$, 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. By implementing these constructions in Python code, we can then simulate and analyze such systems. Besides that, simulation provides ample motivation for why and how we deal with queueing systems and is useful for analyzing realistic systems, as mathematical models have severe shortcomings in such cases. Consider, for example, the service process at a check-in desk of KLM. Business customers and economy customers are served by two separate queueing systems. The business customers are served by one server, server A say, while the economy class customers by three servers, say. What would happen to the sojourn time of the business customers if server A would be allowed to serve economy class customers when the business queue is empty? For the analysis of such complicated control policies, simulation is the most natural approach.

In Chapter 2 and Chapter 3 we derive exact and approximate models, respectively, for single-station queueing systems. The benefit of such models is that they offer structural insights into the behavior of the system and scaling laws, such as that the average waiting time scales (more or less) linearly in the variance of the service times of individual customers. The main idea 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. In fact, the performance measures defined for sample paths are precisely those we compute with simulation.

In Chapter 4 we construct algorithms to analyze open and closed queueing networks. Many of the sample path results developed for the single-station case can be applied to these networks. Thus, via sample-path methods we relate the theory, algorithms, and simulation of queueing systems. For this part we refer to the book of Prof. Zijm; the present set of notes augments the discussion there.

Our aim is not to provide rigorous proofs for all results derived in the book. For this we refer to the following books.

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

EXERCISES

I urge you to try to make as many exercises as possible. The main text contains hardly any examples or derivations: the exercises *illustrate* the material and force you to *think* about the technical parts. The exercises require many of the tools you learned previously in courses on calculus, probability, and linear algebra. Here you can see them applied. Moreover, many of these tools will be useful for other, future, courses. Thus, the investments made here will pay off for the rest of your (student) career.

You'll notice that some of these problems are quite difficult, often not because the problem itself is difficult, but because you need to combine a substantial amount of knowledge all at the same time. All this takes time and effort. Realize that the exercises are not intended to be easy (otherwise we could have been satisfied with computing $1+1$).

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. The companion document also contains many additional simple exercises and old exam questions.

Exercises marked as 'not obligatory' are interesting, but 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. Prof. dr. W.H.M. Zijm allowed me to use the first few chapters of his book. 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.

1

CONSTRUCTION AND SIMULATION OF QUEUEING SYSTEMS

In this chapter we start with a discussion of the Poisson process. We then construct queueing processes in discrete time and apply the Poisson process to model the number of arrivals in periods of fixed length. In Section 1.4 we relate the exponential distribution to the Poisson distribution. The exponential distribution often serves as a good model for inter-arrival times of individual jobs. As such, this is a key component of the construction of queueing processes in continuous time. As it turns out, both ways to construct queueing processes are easily implemented as computer programs, thereby allowing us to use simulation to analyze queueing systems. In passing, we develop a number of performance measures to provide insight into the (transient and average) behavior of queueing processes. We then introduce a useful set of shorthands to distinguish between different queueing models, and finish the chapter with a motivation why we focus on a steady-state analysis of queueing systems in the remainder of the book.

We assume that you *know all* results of Section 1.1.

1.1 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 course.

We use the notation:

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

where the last equation defines an *indicator variable*.

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

1.1.1. *[Companion 1.1.1] Let c be a constant (in $\mathbb{R}$) and the functions f and g both of $o(h)$. Then show that (1) $f(h) \to 0$ when $h \to 0$, (2) $c \cdot f = o(h)$, (3) $f + g = o(h)$, and (4) $f \cdot g = o(h)$.*

You should know that:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^{n-i} b^i, \tag{1.1.1a}$$

$$e^x = \lim_{n \to \infty} (1 + x/n)^n, \tag{1.1.1b}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!}, \tag{1.1.1c}$$

$$\sum_{n=0}^{N} \alpha^n = \frac{1-\alpha^{N+1}}{1-\alpha}. \tag{1.1.1d}$$

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

$$X = \sum_{n=0}^{\infty} X\,\mathbb{1}_{X=n} = \sum_{n=0}^{\infty} n\,\mathbb{1}_{X=n}, \tag{1.1.2a}$$

$$\mathsf{E}[X] = \sum_{n=0}^{\infty} n f(n), \tag{1.1.2b}$$

$$\mathsf{E}[g(X)] = \sum_{n=0}^{\infty} g(n) f(n), \tag{1.1.2c}$$

For general random variables X and Y:

$$\mathsf{E}[\mathbb{1}_{X\le x}] = \mathsf{P}(X \le x), \tag{1.1.3}$$

$$\mathsf{V}[X] = \mathsf{E}\left[X^2\right] - (\mathsf{E}[X])^2. \tag{1.1.4}$$

$$\mathsf{E}[X+Y] = \mathsf{E}[X] + \mathsf{E}[Y], \tag{1.1.5}$$

$$\mathsf{V}[X+Y] = \mathsf{V}[X] + \mathsf{V}[Y], \quad \text{if } X \text{ and } Y \text{ are independent.} \tag{1.1.6}$$

1.1.2. *[Companion 1.1.4] Define the* survivor function *of X as $G(k) = \mathsf{P}(X > k)$. 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.

1.1.3. *[Companion 1.1.5] Express the probability mass $f(k)$ and the survivor function $G(k)$ in terms of the* distribution function *$F(k) = \mathsf{P}(X \le k)$ of X.*

1.1.4. *[Companion 1.1.8] Use indicator functions to prove that $\sum_{i=0}^{\infty} iG(i) = \mathsf{E}\left[X^2\right]/2 - \mathsf{E}[X]/2$.*

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

$$\mathsf{E}[X] = \int_0^{\infty} x\,\mathrm{d}F(x)$$

for the expectation of X. Here $\mathrm{d}F(x)$ acts as a shorthand for $f(x)\mathrm{d}x$[1]. Recall that

$$\mathsf{E}[g(X)] = \int_0^{\infty} g(x)\,\mathrm{d}F(x).$$

1.1.5. *[Companion 1.1.9] Use indicator functions to prove that $\mathsf{E}[X] = \int_0^{\infty} x\,dF(x) = \int_0^{\infty} G(y)\,dy$, where $G(x) = 1 - F(x)$.*

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

1.1.6. *[Companion 1.1.11] Show that $\mathsf{E}\left[X^2\right]/2 = \int_0^{\infty} yG(y)\,dy$ for a continuous non-negative random variable X with survivor function G.*

[1] For the interested reader, $\int x\,\mathrm{d}F(x)$ is a Lebesgue-Stieltjes integral with respect to the distribution function F.

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

$$M_X(s) = \mathsf{E}\left[e^{sX}\right], \tag{1.1.7a}$$

$$M_X(s) \text{ uniquely characterizes the distribution of } X, \tag{1.1.7b}$$

$$\mathsf{E}[X] = M_X'(0) = \left.\frac{\mathrm{d}M_X(s)}{ds}\right|_{s=0}, \tag{1.1.7c}$$

$$\mathsf{E}\left[X^2\right] = M_X''(0), \tag{1.1.7d}$$

$$M_{X+Y}(s) = M_X(s) \cdot M_Y(s), \quad \text{if } X \text{ and } Y \text{ are independent.} \tag{1.1.7e}$$

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

1.1.7. *[Companion 1.1.13] 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:

$$\mathsf{P}(A \mid B) = \frac{\mathsf{P}(AB)}{\mathsf{P}(B)}, \quad \text{if } \mathsf{P}(B) > 0, \tag{1.1.8a}$$

$$\mathsf{P}(A) = \sum_{i=1}^{n} \mathsf{P}(AB_i) = \sum_{i=1}^{n} \mathsf{P}(A \mid B_i)\mathsf{P}(B_i), \quad \text{if } A = \bigcup_{i=1}^{n} B_i \text{ and } \mathsf{P}(B_i > 0) \text{ for all } i. \tag{1.1.8b}$$

1.2 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.

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 λ such that

$$\mathsf{E}[N(s,t]] = \lambda(t-s). \tag{1.2.1}$$

The constant λ 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 subintervals, 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 such that $p = \mathsf{P}(B_i = 1)$ and $\mathsf{P}(B_i = 0) = 1 - \mathsf{P}(B_i = 1)$, and we assume that B_i and B_j are independent whenever $i \neq j$. The total number of arrivals $N_n(t)$ that occur in n intervals is then *binomially distributed*, i.e.,

$$\mathsf{P}(N_n(t) = k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{1.2.2}$$

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

$$\mathsf{P}(N(t) = k) = e^{-\lambda t} \frac{(\lambda t)^k}{k!}, \tag{1.2.3}$$

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

We call the process $N_\lambda = \{N(t)\}$ a *Poisson process* with rate λ when N_λ is stationary, has independent increments, and its elements $N(t) \sim P(\lambda t)$ for all t. Observe that the process N_λ 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.

1.2.1. *[Companion 1.2.4] Show that*

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

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

1.2.2. *[Companion 1.2.8] Show that if $N(t) \sim P(\lambda t)$, the expected number of arrivals during $[0,t]$ is*

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

1.2.3. *[Companion 1.2.11] Use the moment-generating function of $N(t) \sim P(\lambda t)$ to compute $\mathsf{E}[N(t)]$ and $\mathsf{V}[N(t)]$.*

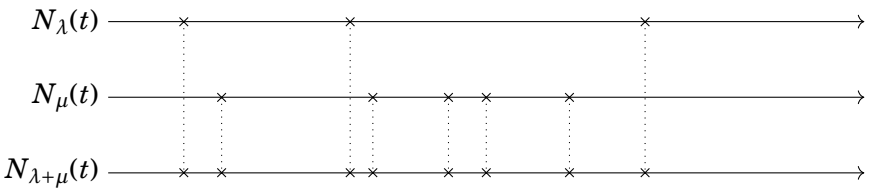
Define the *square coefficient of variation* (*SCV*) of a random variable X as

$$C^2 = \frac{\mathsf{V}[X]}{(\mathsf{E}[X])^2}. \tag{1.2.4}$$

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

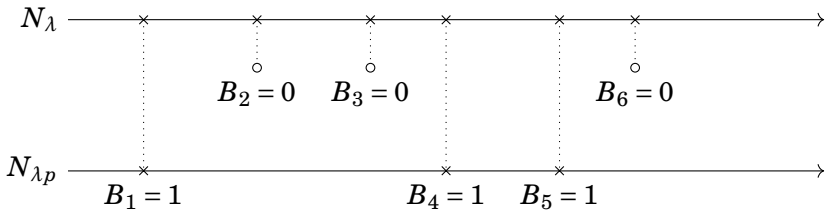
1.2.4. *[Companion 1.2.12] Show that the SCV of $N(t) \sim P(\lambda t)$ is equal to $1/(\lambda t)$. What does this mean for t large?*

Merging Poisson processes occurs often in practice. We have two Poisson processes, for instance, the arrival processes N_λ of men and N_μ 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 $\lambda + \mu$.



1.2.5. *[Companion 1.2.13] If the Poisson arrival processes N_λ and N_μ are independent, show with a conditioning argument that $N_\lambda + N_\mu$ is a Poisson process with rate $\lambda + \mu$.*

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_λ. 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.



1.2.6. *[Companion 1.2.17] Show with moment-generating functions that thinning the Poisson process N_λ 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.

1.2.7. *[Companion 1.2.18] Use moment-generating functions to prove that $N_n(t)$ converges to N, that is, the right-hand side in (1.2.2) converges to the Poisson distribution (1.2.3) when $n \to \infty, p \to 0$ such that $pn = \lambda t$ remains constant.*

1.3 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

$$d_k = \min\{L_{k-1}, c_k\}, \tag{1.3.1a}$$

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

$$L_k = L_{k-1} - d_k + a_k. \tag{1.3.1b}$$

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.

1.3.1. *[Companion 1.3.3] Show that the scheme*

$$\begin{aligned} L_k &= [L_{k-1} + a_k - c_k]^+, \\ d_k &= L_{k-1} + a_k - L_k. \end{aligned} \tag{1.3.2}$$

is equivalent to a modification of (1.3.1) *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 $\{a_k\}_{k=1,2,\ldots}$ and the capacity $\{c_k\}_{k=1,2,\ldots}$ and feed this data into a computer to carry out the recursions (1.3.1). 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 $a_1, \ldots, a_n$ 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 1.2. 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:

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

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 $C_{1,1} = C_{2,2} = \cdots = C_{1,6} = C_{2,7} = \cdots = 0$, 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., $C_{i,5} = C_{i,10} = \cdots = 0$ 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 (1.3.1) 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 intakes less (in other, 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.

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.

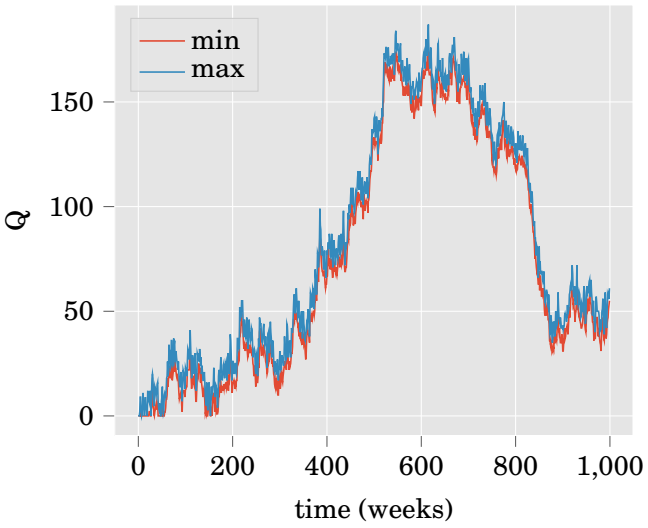


Figure 1: Effect of capacity and holiday plans. We plot for each time point the maximum and the minimum queue length for each of the policies. Apparently, the effect of each of these policies is, for all practical purposes, negligible.

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[2]. 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

[2] 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.

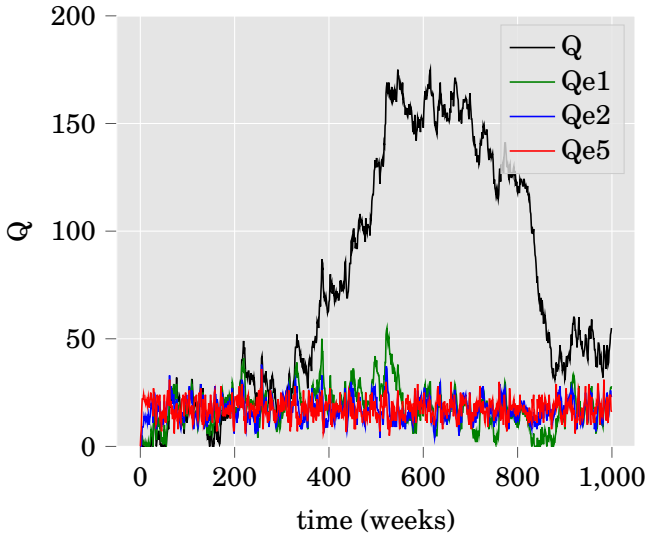


Figure 2: Controlling the number of intakes. Clearly, adapting the service rate 'does wonders' to control the queue length.

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.

1.3.2. *[Queue with Blocking][Companion 1.3.4] 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?*

1.3.3. *[Estimating the lead time distribution] [Companion 1.3.5] 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 $W_{+,k}$.*

1.3.4. *[Cost models][Companion 1.3.8] 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 β 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.*

1.3.5. *[N-policies][Companion 1.3.9] 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 β 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.*

1.3.6. *[Priority queueing][Companion 1.3.12] 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.*

1.3.7. *[Queue with protected service capacity and lost capacity][Companion 1.3.14] 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.

1.3.8. *[Tandem networks][Companion 1.3.15] 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 (1.3.1) to simulate this situation.*

1.3.9. *[Merging departure streams][Companion 1.3.17] 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.*

1.4 EXPONENTIAL DISTRIBUTION

In Section 1.2 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*.

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

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

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 λ if and only if the inter-arrival times $\{X_i\}$ are *i.i.d.* (*independent and identically distributed*) and exponentially distributed with mean $1/\lambda$, 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.

1.4.1. *[Companion 1.4.2] If the random variable $X \sim \text{Exp}(\lambda)$, show that its mean $\mathsf{E}[X] = \frac{1}{\lambda}$.*

1.4.2. *[Companion 1.4.7] Use the moment-generating function of $X \sim \text{Exp}(\lambda)$ to show that*

$$\mathsf{E}[X] = \frac{1}{\lambda}, \qquad \mathsf{E}\left[X^2\right] = \frac{2}{\lambda^2}.$$

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

1.4.3. *[Companion 1.4.9] If N_λ is a Poisson process with rate λ, show that the time X_1 to the first arriving job is* $\text{Exp}(\lambda)$.

1.4.4. *[Companion 1.4.11] 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*

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

1.4.5. *[Companion 1.4.13] 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

$$\mathsf{P}(X > t+h | X > t) = \mathsf{P}(X > h).$$

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.

1.4.6. *[Companion 1.4.14] 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].

1.4.7. *[Companion 1.4.15] 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 $\mathsf{E}[Z] = (\lambda+\mu)^{-1}$.

1.4.8. *[Companion 1.4.16] If $X \sim \text{Exp}(\lambda)$, $S \sim \text{Exp}(\mu)$, and X and S are independent, show that*

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

1.5 CONSTRUCTION OF THE SINGLE-SERVER QUEUEING PROCESS IN CONTINUOUS TIME

In Section 1.3 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 (1.3.1) 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.

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 kth job must lie somewhere in $(s,t]$. Thus, from $\{A(t)\}$, we can define[3]

$$A_k = \min\{t : A(t) \geq k\}, \tag{1.5.1}$$

[3] 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.

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

$$X_k = A_k - A_{k-1}. \tag{1.5.2}$$

However, often the basic data consists of the inter-arrival times $\{X_k; k = 1,2,\ldots\}$ 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

$$A(t) = \max\{k : A_k \le t\}. \tag{1.5.3}$$

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 $\{X_k\}$.

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

$$A(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t}. \tag{1.5.4}$$

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

$$W_{Q,k} = [W_{Q,k-1} + S_{k-1} - X_k]^+. \tag{1.5.5}$$

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.

1.5.2. *[Companion 1.5.6] 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 $\mathsf{P}(S - X \le u)$, 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

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

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

$$W_k = D_k - A_k = \tilde{A}_k + S_k - A_k = W_{Q,k} + S_k, \tag{1.5.6}$$

where each of these equations has its own interpretation.

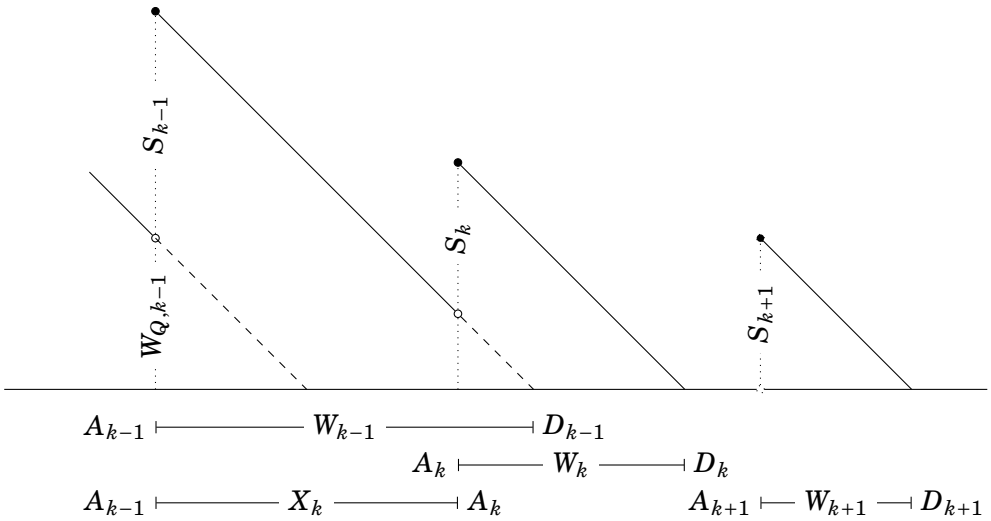


Figure 3: Construction of the single-server queue in continuous time. The sojourn time W_k of the kth job is the sum of the work in queue $W_{Q,k}$ at its arrival epoch A_k and its service time S_k; its departure time is then $D_k = A_k + W_k$. The waiting time of job k is clearly equal to $W_{k-1} - X_k$. 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.

1.5.3. *[Companion 1.5.11] Explain the following recursions for a single server queue:*

$$\begin{aligned} A_k &= A_{k-1} + X_k, \\ D_k &= \max\{A_k, D_{k-1}\} + S_k, \\ W_k &= D_k - A_k. \end{aligned} \tag{1.5.7}$$

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.

1.5.4. *[Companion 1.5.12] 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,

$$L(t) = A(t) - D(t) + L(0), \tag{1.5.8}$$

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 $L_s(t)$.

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.

1.5.5. *[Companion 1.5.18] Show that $L(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t < D_k}$ when the system starts empty.*

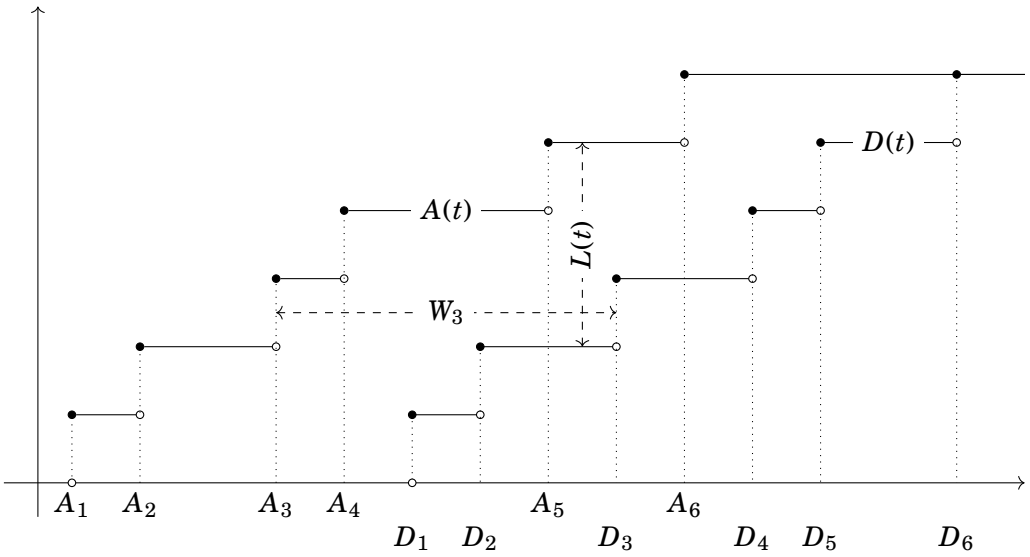


Figure 4: Relation between the arrival process $\{A(t)\}$, the departure process $\{D(t)\}$, the number in the system $\{L(t)\}$ and the waiting times $\{W_k\}$. In particular, $L(t)$ is the difference between the graphs of $A(t)$ and $D(t)$.

Define the number of jobs in the system as seen by the kth arrival as

$$L(A_k-). \tag{1.5.9}$$

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.

1.5.6. *[Companion 1.5.20] With the recursions* **??** *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:*

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

1.6 KENDALL'S NOTATION

As became apparent in Sections 1.3 and 1.5, 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.

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.[4] In this notation it is

[4] 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.

assumed that jobs are served in first-in-first-out (FIFO) order; FIFO scheduling is also often called first-come-first-served (FCFS).

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.

1.6.1. *[Companion 1.6.7] What is the meaning of $M(n)/M(n)/1$?*

1.6.2. *[Companion 1.6.8] What is the meaning of $M^X/M/1$?*

1.6.3. *[Companion 1.6.13] 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.

1.6.4. *[Companion 1.6.14] 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.

1.6.5. *[Companion 1.6.15] 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.*

1.7 QUEUEING PROCESSES AS REGULATED RANDOM WALKS

In the construction of queueing processes as set out in Section 1.3 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., (1.3.2). Observe that in (1.3.2) the process $\{L_k\}$ shares a resemblance to a random walk $\{Z_k, k = 0,1,\ldots\}$ with Z_k given by

$$Z_k = Z_{k-1} + a_k - c_k. \tag{1.7.1}$$

To see that $\{Z_k\}$ is indeed a random walk, observe that Z makes jumps of size $a_k - c_k, k = 1,\ldots,$ 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. In this section, we show how to build the queueing process $\{L_k\}$ from the random walk $\{Z_k\}$ using a device called a *reflection map*, which gives an elegant construction of a queueing process. Moreover, we can use the probabilistic tools that have been developed for the random walk to analyze queueing systems. One example is the distribution of the time until an especially large queue length is reached; these times can be formulated as *hitting times* of the random walk. Another example is the average time it takes to clear a large queue.

1.7.1. *[Companion 1.7.1] Show that L_k satisfies the relation*

$$L_k = Z_k - \min_{1\le i\le k} Z_i \wedge 0, \tag{1.7.2}$$

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., $\mathsf{P}(a_k = 1) = 0.3 = 1 - \mathsf{P}(a_k = 0)$, and $c_k \sim B(0.4)$. In Fig. 6, $a_k \sim B(0.49)$ and the random walk is constructed as

$$Z_k = Z_{k-1} + 2a_k - 1. \tag{1.7.3}$$

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 (1.7.1); there, $Z_k = Z_{k-1}$, if $a_k = c_k$.

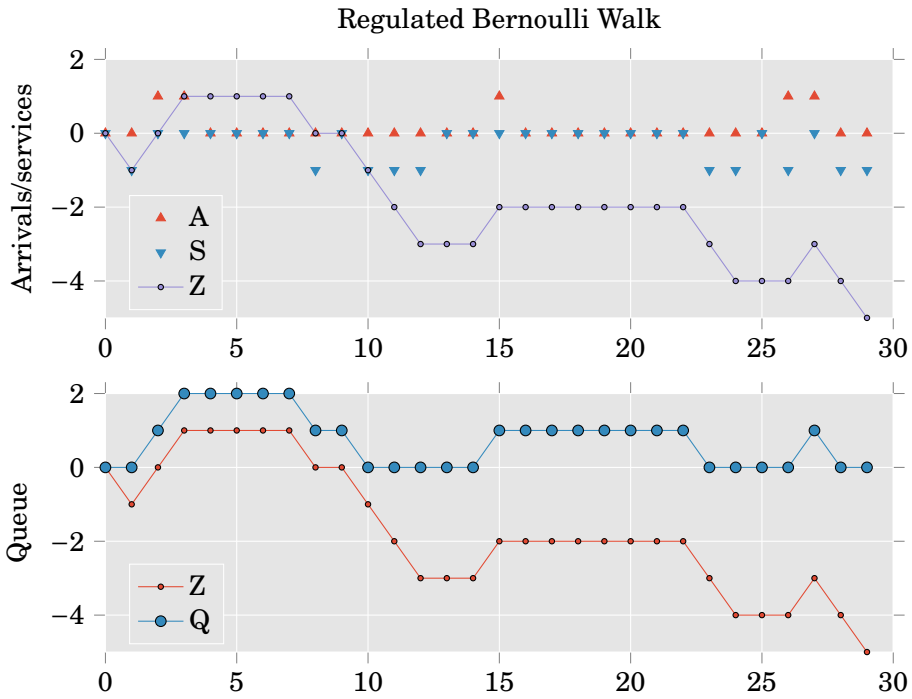


Figure 5: The upper panel shows a graph of the random walk Z. An upward pointing triangle corresponds to an arrival, a downward triangle to a potential service. The lower panel shows the queueing process $\{L_k\}$ as a random walk with reflection.

With (1.7.2), 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 $c_k \sim P(\mu)$.

1.7.2. *[Companion 1.7.2] 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.

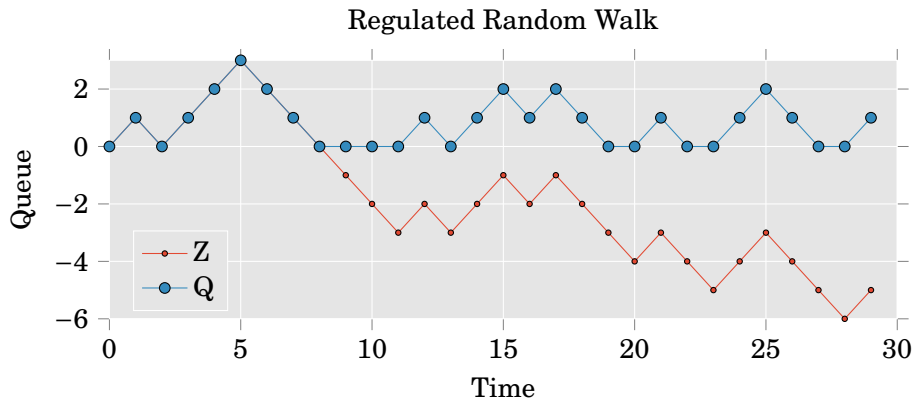


Figure 6: Another example of a reflected random walk.

1.7.3. *[Companion 1.7.3] Show that when $n > m$ and $Z_0 = m$,*

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

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 (1.7.2)), 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 $t \to \infty$. The limiting random variable L is known as the *steady-state limit* of the sequence of random variables $\{L_k\}$, and the distribution of L is known as the *limiting distribution* or *stationary distribution* of $\{L_k\}$. Taking these limits warrants two questions: what type of limit is actually meant here, and what is the rate of convergence to this limiting situation? Here we sidestep all such fundamental issues, as the details require measure theory and more advanced probability theory than we can deal with here.

To provide some intuition about the rate of convergence we consider now an example. Specifically, 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 (1.5.5). 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 (1.5.5) to compute the *exact* distribution of $W_{Q,k}$ for $k = 1,2,\ldots,20$, cf., the left panel in Fig. 7. We see that when $k = 5$, the 'hump' of $\mathsf{P}(W_{Q,5} = x)$ 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. Here the *empirical distribution* for the simulation is defined as

$$\mathsf{P}(W_Q \le x) = \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{W_{Q,k} \le x},$$

where $W_{Q,k}$ is obtained by simulation. As should be clear from the figure, the simulated distribution also converges quite fast to some limiting function. Finally, in the right panel we

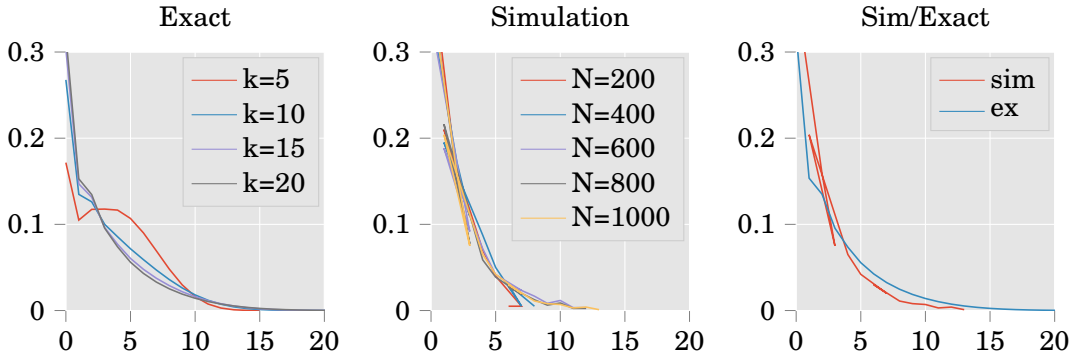


Figure 7: The density of $W_{Q,k}$ for $k = 5, 10, 15, 20$ computed by an exact method as compared the density obtained by simulation of different run lengths $N = 200, 400, \ldots, 1000$. The right panel compares the exact density of $W_{Q,20}$ to the density obtained by simulation for $N = 1000$.

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 *stationarity*. The study of queueing systems in stationary state will occupy us for the rest of the book.

1.7.4. *[Companion 1.7.5] Suppose that* $X_k \in \{1, 3\}$ *such that* $\mathsf{P}(X_k = 1) = \mathsf{P}(X_k = 3)$ *and* $S_k \in \{1, 2\}$ *with* $\mathsf{P}(S_k = 1) = \mathsf{P}(S_k = 2)$. *Write a computer program to see how fast the distributions of* $W_{Q,k}$ *converge to a limiting distribution function.*

1.7.5. *[Companion 1.7.6] Validate the results of Fig. 7 with simulation.*

2

ANALYTICAL MODELS

In this chapter we focus on developing analytic models for various queueing systems in steady-state. In the analysis we use sample-path and level-crossing arguments to count how often certain events occur as a function of time. Then we define probabilities in terms of limits of fractions of these counting processes. Like this the performance measures can be explicitly computed for the statistical analysis of (simulations of) queueing systems.

As a reminder, we keep the discussion in these notes mostly at an intuitive level, and refer to El-Taha and Stidham Jr. [1998] for proofs and further background.

2.1 RATE STABILITY AND UTILIZATION

In the analysis of any queueing process, the first step should be to check the relations between the arrival, service and departure rates. The concept of rate is crucial because it captures our intuition that when, in the long run, jobs arrive faster than they can leave, the system must 'explode'. As we will see, when the arrival rate is smaller than the service rate, the system is stable. Thus, the first performance measure we need to estimate for a queueing system is the ratio between the arrival and service rate. In this section, we develop some concepts and notation to formalize these ideas. We will use these concepts throughout the remainder of the book.

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

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t}. \tag{2.1.1}$$

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 *independent and identically distributed (i.i.d.)* and distributed as a generic random variable X with finite mean $\mathsf{E}[X]$, we can construct $\{A_k\}$ and $\{A(t)\}$ as described in Section 1.5; the strong law of large numbers then guarantees that the above limit exists.

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 $A_n \to \infty$ if $n \to \infty$, it follows from (2.1.1) that the average inter-arrival time between two consecutive jobs is

$$\mathsf{E}[X] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \lim_{n\to\infty} \frac{A_n}{A(A_n)} = \lim_{t\to\infty} \frac{t}{A(t)} = \frac{1}{\lambda}, \tag{2.1.2}$$

where we take $t = A_n$ in the limit for $t \to \infty$. In words, the above states that the arrival rate λ is the inverse of the expected inter-arrival time.

The development of the departure times $\{D_k\}$ is entirely analogous to that of the arrival times; define the *departure rate* as

$$\delta = \lim_{t\to\infty} \frac{D(t)}{t}. \tag{2.1.3}$$

Assume now that there is a single server. Let S_k be the required service time of the kth job to be served, and define

$$U_n = \sum_{k=1}^{n} S_k$$

as the total service time required by the first n jobs. With this, let

$$U(t) = \max\{n : U_n \le t\}$$

and define the *service rate* or *processing rate* as

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

In the same way as we derived that $\mathsf{E}[X] = 1/\lambda$, we obtain for the expected (or average) service time required by an individual job

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

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

$$\delta = \lim_{t\to\infty} \frac{D(t)}{t} \le \lim_{t\to\infty} \frac{A(t)}{t} = \lambda. \tag{2.1.4}$$

We call a system *rate stable* if

$$\lambda = \delta,$$

in other words, the system is stable if, in the long run, jobs leave the system just as fast as they arrive. Of course, if $\lambda > \delta$, the system length process $L(t) \to \infty$ as $t \to \infty$.

It is also evident that jobs cannot depart faster than they can be served, hence, $D(t) \le U(t)$ for all t. Combining this with the fact that $\delta \le \lambda$, we get

$$\delta \le \min\{\lambda, \mu\}.$$

When $\mu \ge \lambda$ the above inequality reduces to $\delta = \lambda$ for rate-stable systems[1]. As it turns out, when $\mu = \lambda$ and the variance of the service time $\mathsf{V}[S] > 0$ or $\mathsf{V}[X] > 0$ the queue length process can behave in a very peculiar way. For this reason we henceforth (and implicitly) require that $\mu > \lambda$.

2.1.1. *[Companion 2.1.5] Define $\tilde{X}_k = S_{k-1} - X_k$. Show that $\mathsf{E}\left[\tilde{X}_k\right] < 0$ implies that $\lambda < \mu$.*

2.1.2. *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, $L(t)/t \to 0$ as $t \to \infty$.*

[1] It would be interesting to prove this.

2.1.3. *[Companion 2.1.7] 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 λ_A for type A and λ_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 (rate) 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, so that $\mu < \lambda$. Thus, it is necessary to identify minimal batch sizes to ensure that $\mu > \lambda$. Motivate that the following linear program can be used to determine the minimal batch sizes:

$$\textit{minimize } T$$

such that $T = k_A t_A + S + k_B t_B$, $\lambda_A T < k_A$ and $\lambda_B T < k_B$.

2.2 RENEWAL REWARD THEOREM AND LOAD

We start with stating and proving (graphically) the *renewal reward theorem*. In the sequel, we will see many applications of this theorem. In this section, we use it to relate the fraction of time the server is busy in a $G/G/1$ queue to the job arrival rate and the expected job service time.

The renewal reward theorem is very useful and states intuitively that when customers arrive at rate λ and each customer pays an average amount X, then the system earns money at rate $Y = \lambda X$. Figure 8 provides graphical motivation about why this theorem is true; El-Taha and Stidham Jr. [1998] gives a (simple) proof.

Theorem 2.2.1 (Renewal Reward Theorem, $Y = \lambda X$)**.** Consider epochs $\{T_k, k = 0, 1, \ldots\}$ such that $0 = T_0 < T_1 < \cdots$ 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 $N(t)/t \to \lambda$ as $t \to \infty$, 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,

$$\lim_{t\to\infty} \frac{Y(t)}{t} = Y \iff \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} X_k = X,$$

and then $Y = \lambda X$.

Define the *load* or *utilization* as the limiting fraction of time the server is busy, i.e.,

$$\rho = \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathbb{1}_{L(s)>0}\, \mathrm{d}s.$$

2.2.1. *[Companion 2.2.1] Use the renewal reward theorem to prove that $\rho = \lambda \mathsf{E}[S]$ for the rate-stable $G/G/1$ queue.*

2.2.2. *[Companion 2.2.2] We can derive the relation $\rho = \lambda \mathsf{E}[S]$ in a somewhat more direct way by considering the fact that*

$$\sum_{k=1}^{A(t)} S_k \geq \int_0^t \mathbb{1}_{L(s)>0}\, ds \geq \sum_{k=1}^{D(t)} S_k.$$

Explain this, and complete the argument.

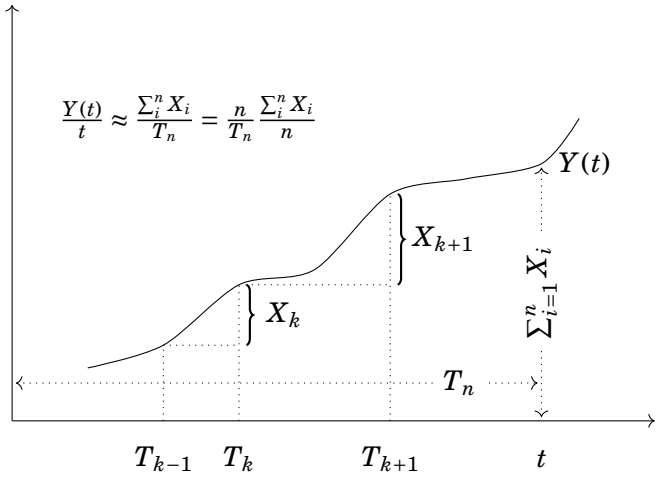


Figure 8: A graphical 'proof' of $Y = \lambda X$. Here $Y(t)/t \to Y$, $n/T_n \to \lambda$ and $n^{-1}\sum_i^n X_i \to X$. Observe that in the figure X_k does not represent an inter-arrival time; instead it corresponds to the increment of (the graph of) $Y(t)$ between two consecutive epochs T_{k-1} and T_k at which $Y(t)$ is observed.

From the identities $\lambda^{-1} = \mathsf{E}[X]$ and $\mu^{-1} = \mathsf{E}[S]$, we get a further set of relations:

$$\rho = \lambda\,\mathsf{E}[S] = \frac{\lambda}{\mu} = \frac{\mathsf{E}[S]}{\mathsf{E}[X]}.$$

Thus, the load has also the interpretation as the rate at which jobs arrive times the average amount of work per job. Finally, recall that for a system to be rate-stable, it is necessary that $\mu > \lambda$, implying in turn that $\rho < 1$. The relation $\rho = \mathsf{E}[S]/\mathsf{E}[X] < 1$ 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., $\mathsf{E}[S] < \mathsf{E}[X]$. In fact, when $\mu < \lambda$, it is easy to check with simulation that the queue length grows roughly linearly with slope $\lambda - \mu$.

2.2.3. *[Companion 2.2.3] Consider a queueing system with c servers with identical production rates μ. What would be a reasonable stability criterion for this system?*

2.3 (LIMITS OF) EMPIRICAL PERFORMANCE MEASURES

If the arrival and service processes are such that the queueing system is rate-stable, we can sensibly define other performance measures such as the average waiting time. In this section, we define the second most important performance measures; recall that the most important is the utilization ρ. At the end we provide an overview of the relations between these performance measures in Fig. 16.

With the construction of queueing processes in Section 1.5 we can compute the waiting time as observed by the first n, say, jobs. We therefore define the *expected waiting time* as

$$\mathsf{E}[W] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} W_k, \tag{2.3.1}$$

and the expected time in queue as

$$\mathsf{E}\left[W_Q\right] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} W_{Q,k}. \tag{2.3.2}$$

Note that these performance measures are limits of *empirical* measures. Note also that these statistics are as *observed by arriving jobs*: the first job has a waiting time W_1 at its arrival epoch, the second a waiting time W_2, and so on. For this reason, we colloquially say that $\mathsf{E}[W]$ is the average waiting time as 'seen by arrivals'. The *distribution of the waiting times at arrival times* can be found by counting:

$$\mathsf{P}(W \leq x) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{W_k \leq x}. \tag{2.3.3}$$

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

$$\mathsf{E}[L] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} L(A_k-), \tag{2.3.4}$$

where $L(A_k-)$ is the number of jobs in the system at the arrival epoch of the kth job. The *distribution of* $\{L(t)\}$ *as seen by customers upon arrival*, is

$$\mathsf{P}(L \leq m) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{L(A_k-)\leq m}. \tag{2.3.5}$$

We call $\mathsf{P}(L > m)$ the *excess probability*.

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 sampling (observation) moments. Assuming the limit exists we use (1.5.8) to define the *time-average number of jobs* as

$$\mathsf{E}[L] = \lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)\,\mathrm{d}s. \tag{2.3.6}$$

Observe that, notwithstanding that the symbols are the same, this expectation need not be the same as (2.3.4). In a loose sense we can say that $\mathsf{E}[L]$ 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

$$\mathsf{P}(L \leq m) = \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathbb{1}_{L(s)\leq m}\,\mathrm{d}s. \tag{2.3.7}$$

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

2.3.1. *[Companion 2.3.1] 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 the customers see.*

2.3.2. *[Companion 2.3.3] Consider a discrete-time model of a queueing system, such as the ones developed in Section 1.3. 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 one definition similar to (2.3.5) for the case in which we say that all arrivals see the same number in the system. Provide a second in which we like to express that the first of a batch of arrivals sees less in the system than the last arrival of a batch.

2.4 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 must remain approximately the same, 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 single-server queueing systems.

To establish the section's main result (2.4.5) 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. Note that Figure 17 at the end of the chapter summarizes all concepts we develop here.

LEVEL CROSSING Define

$$A(n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t} \mathbb{1}_{L(A_k-)=n} \tag{2.4.1a}$$

as the number of arrivals up to time t that saw n customers in the system upon their arrival, cf. Fig. 9.

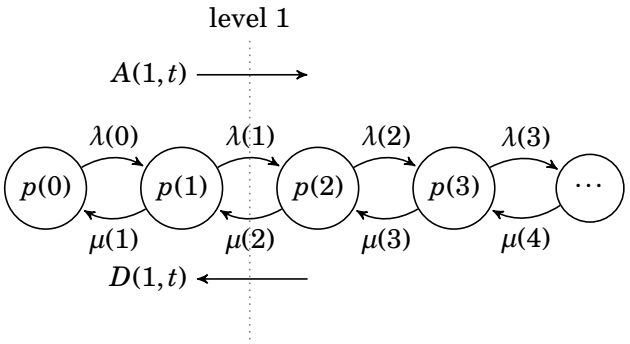


Figure 9: $A(1,t)$ counts the number of jobs up to time t that saw 1 job in the system upon arrival, and right after such arrivals the system contains 2 jobs. Thus, each time $A(1,t)$ increases by one, level 1 (the dotted line separating states 1 and 2) is crossed from below. Similarly, $D(1,t)$ counts 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.)

Next, let

$$Y(n,t) = \int_0^t \mathbb{1}_{L(s)=n} \, \mathrm{d}s \tag{2.4.1b}$$

be the total time the system contains n jobs during $[0,t]$, and

$$p(n,t) = \frac{1}{t} \int_0^t \mathbb{1}_{L(s)=n} \, \mathrm{d}s = \frac{Y(n,t)}{t}, \tag{2.4.1c}$$

be the fraction of time that $L(s) = n$ in $[0,t]$. Figure 10 illustrates the relation between $Y(n,t)$ and $A(n,t)$.

2.4.1. *[Companion 2.4.5] Consider the following (silly) queueing process. At times* $0,2,4,\ldots$ *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?)*

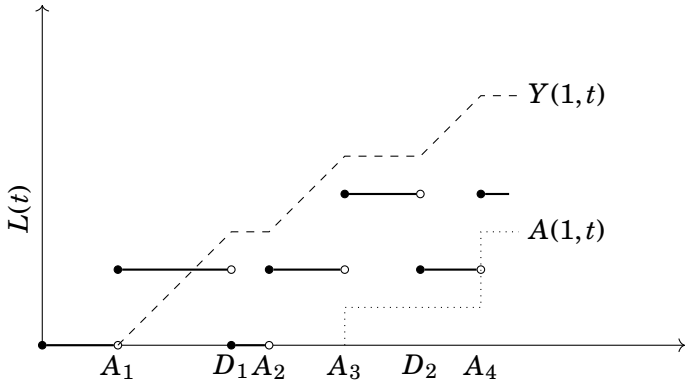


Figure 10: Plots of $Y(1,t)$ and $A(1,t)$. (For visual clarity, we subtracted 1/2 from $A(1,t)$, for otherwise its graph would partly overlap with the graph of Y.)

2.4.2. *[Continuation of 2.4.1] [Companion 2.4.6] Find an expression for $Y(n,t)$.*

Define also the limits:

$$\lambda(n) = \lim_{t\to\infty} \frac{A(n,t)}{Y(n,t)}, \qquad p(n) = \lim_{t\to\infty} p(n,t), \tag{2.4.2}$$

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.

2.4.3. *[Continuation of 2.4.2] [Companion 2.4.7] Compute $p(n)$ and $\lambda(n)$.*

Similar to the definition for $A(n,t)$, let

$$D(n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \le t}\, \mathbb{1}_{L(D_k)=n}$$

denote the number of departures up to time t that *leave n customers behind*. Then, define

$$\mu(n+1) = \lim_{t\to\infty} \frac{D(n,t)}{Y(n+1,t)},$$

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 9 shows how $A(n,t)$ and $\lambda(n)$ relate to $D(n+1,t)$ and $\mu(n)$.

2.4.4. *[Continuation of 2.4.3] [Companion 2.4.9] Compute $D(n,t)$ and $\mu(n+1)$ for $n \ge 0$.*

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 $|A(n,t) - D(n,t)| \le 1$ (think about this). From this observation it follows immediately that

$$\lim_{t\to\infty} \frac{A(n,t)}{t} = \lim_{t\to\infty} \frac{D(n,t)}{t}. \tag{2.4.3}$$

With this equation we can obtain two nice and fundamental identities. The first we develop now; the second follows in Section 2.7.

The rate of jobs that 'see the system with n jobs' can be defined as $A(n,t)/t$. Taking limits we get

$$\lim_{t\to\infty}\frac{A(n,t)}{t}=\lim_{t\to\infty}\frac{A(n,t)}{Y(n,t)}\frac{Y(n,t)}{t}=\lambda(n)p(n), \tag{2.4.4a}$$

where we use the above definitions for $\lambda(n)$ and $p(n)$. Similarly, the departure rate of jobs that leave n jobs behind is

$$\lim_{t\to\infty}\frac{D(n,t)}{t}=\lim_{t\to\infty}\frac{D(n,t)}{Y(n+1,t)}\frac{Y(n+1,t)}{t}=\mu(n+1)p(n+1). \tag{2.4.4b}$$

Combining this with (2.4.3) we arrive at the *level-crossing equations*

$$\lambda(n)p(n)=\mu(n+1)p(n+1). \tag{2.4.5}$$

2.4.5. *[Continuation of 2.4.4][Companion 2.4.10] Compute $\lambda(n)p(n)$ for $n\geq 0$, and check $\lambda(n)p(n)=\mu(n+1)p(n+1)$.*

Result (2.4.5) turns out to be exceedingly useful, as will become evident from Section 2.5 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

$$p(n+1)=\frac{\lambda(n)}{\mu(n+1)}p(n). \tag{2.4.6}$$

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

$$p(n+1)=\frac{\lambda(n)\lambda(n-1)\cdots\lambda(0)}{\mu(n+1)\mu(n)\cdots\mu(1)}p(0). \tag{2.4.7}$$

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*

$$G=1+\sum_{n=0}^{\infty}\frac{\lambda(n)\lambda(n-1)\cdots\lambda(0)}{\mu(n+1)\mu(n)\cdots\mu(1)}. \tag{2.4.8}$$

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 (2.4.5), 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

$$\mathsf{E}[L]=\sum_{n=0}^{\infty}np(n),$$

and the long-run fraction of time the system contains at least n jobs is

$$\mathsf{P}(L\geq n)=\sum_{i=n}^{\infty}p(i).$$

2.4.6. *[Companion 2.4.11] Derive $\mathsf{E}[L]=\sum_{n=0}^{\infty}np(n)$ from (2.3.6).*

Finally, the following two exercises show that level-crossing arguments extend well beyond the queueing systems modeled by Fig. 9.

2.4.7. *[Companion 2.4.12] 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 λ 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.*

2.4.8. *[Companion 2.4.13] Use the graph of 2.4.7 and a level-crossing argument to express the steady-state probabilities $p(n), n = 0, \ldots, 3$ in terms of λ and μ.*

INTERPRETATION The definitions in (2.4.1) 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 (2.4.1a) 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

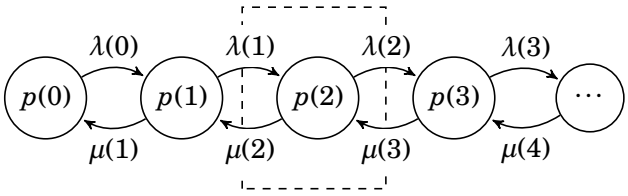


Figure 11: For the balance equations we count how often a box around a state is crossed from inside and outside. In the long run, the entering and leaving rates should be equal. For the example here, the rate out is $p(2)\lambda(2)+p(2)\mu(2)$ while the rate in is $p(1)\lambda(1)+p(3)\mu(3)$.

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. 11. 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, $|I(n,t) - O(n,t)| \le 1$. Thus, from the fact that

$$\lim_{t\to\infty} \frac{I(n,t)}{t} = \lim_{t\to\infty} \frac{A(n-1,t)}{t} + \lim_{t\to\infty} \frac{D(n,t)}{t} = \lambda(n-1)p(n-1) + \mu(n+1)p(n+1)$$

and

$$\lim_{t\to\infty} \frac{O(n,t)}{t} = \lim_{t\to\infty} \frac{A(n,t)}{t} + \lim_{t\to\infty} \frac{D(n-1,t)}{t} = \lambda(n)p(n) + \mu(n)p(n)$$

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 $n \ge 0$ 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.

2.5 $M/M/1$ 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 (2.4.6) we derive a number of important results for this queueing process.

Recall from Section 1.7 that we can construct the $M/M/1$ queue as a reflected random walk where the arrivals are generated by a Poisson process $N_\lambda(t)$ and the departures (provided the number $L(t)$ in the system is positive) are generated according to the Poisson process $N_\mu(t)$. 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 $n \ge 0$ and $\mu(n) = \mu$ for all $n \ge 1$. Thus, (2.4.6) 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 (2.4.8) and (1.1.1d) that

$$p(0) = 1-\rho, \qquad p(n) = (1-\rho)\rho^n. \tag{2.5.1}$$

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,

$$\mathsf{E}[L] = \frac{\rho}{1-\rho}, \qquad \mathsf{V}[L] = \frac{\rho}{(1-\rho)^2}, \qquad \mathsf{P}(L > n) = \rho^{n+1}. \tag{2.5.2}$$

2.5.1. *[Companion 2.5.3] Derive* (2.5.2) *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* $|\rho| < 1$.

Let us interpret (2.5.2). The fact that $\mathsf{E}[L] \sim (1-\rho)^{-1}$ for $\rho \to 1$ implies that the average waiting time increases very fast when $\rho \to 1$. 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 ρ 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 $\mathsf{P}(L \geq 10) = \rho^{10}$ is an estimator for the fraction of customers lost.

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. 12. 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.

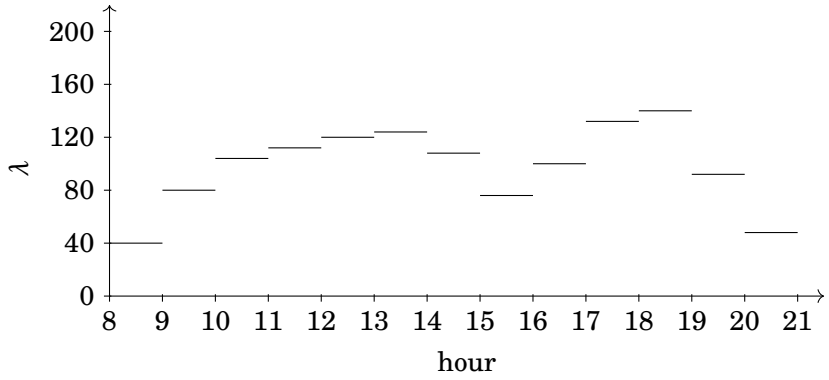


Figure 12: A demand profile of the arrival rate λ modeled as constant over each hour.

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 2.6.3 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 $\mathsf{P}(L > 10) = \rho^{11}$. Combining this with the objective $\mathsf{P}(L > 10) \leq 1\%$, we get that $\rho^{11} \leq 0.01$, which translates into $\rho \leq 0.67$. Using that $\rho = \lambda \mathsf{E}[S]/c$ and our estimate $\mathsf{E}[S] = 1.5$ minutes, we get the following rough bound on c:

$$c \geq \frac{\lambda \mathsf{E}[S]}{0.67} \approx \frac{3}{2} \cdot \lambda \cdot 1.5 = 2.25\lambda,$$

where λ 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. 12 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 λ_t is the demand for hour t.

2.6 $M(n)/M(n)/1$ 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 (2.4.6). For these queueing systems, we just present the results. In the exercises we ask you to derive the formulas—the main challenge is not to make computational errors.

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,

$$\mathsf{P}\left(A_{A(t)+1} \leq t+s \,\middle|\, L(t)=n\right) = 1 - e^{-\lambda(n)s},$$

where we use that $A_{A(t)} + g$ is the arrival time of the next job after time t. Similarly, we assume for all t and s,

$$\mathsf{P}\left(D_{D(t)+1} \leq t+s \,\middle|\, L(t)=n\right) = 1 - e^{-\mu(n)s}.$$

2.6.1. *[Companion 2.6.1] 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.*

2.6.2. *[Companion 2.6.3] Model the $M/M/c$ queue in terms of an $M(n)/M(n)/1$ queue and compute $\mathsf{E}[L_Q]$.*

2.6.3. *[Companion 2.6.5] 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 $M/M/q$ 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 $\mathsf{E}[L]$ and $\mathsf{E}[L_Q]$ of both models as a function of ρ. Explain why these ratios become 1 as $\rho \uparrow 1$.

2.6.4. *[Companion 2.6.6] 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.*

2.6.5. *[Companion 2.6.7] Take the limit $c \to \infty$ 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.*

2.6.6. *[Companion 2.6.12] 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 $N \to \infty$? Interpret the results.*

2.6.7. *[Companion 2.6.13] Give an example of a system with a finite calling population.*

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 2.7 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 $\mathsf{P}(L \le n)$ 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 2.7.

2.6.8. *[Companion 2.6.15] 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?*

2.7 POISSON ARRIVALS SEE TIME AVERAGES

Suppose the following limit exists:

$$\pi(n) = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}_{L(A_k-)=n}, \tag{2.7.1}$$

then $\pi(n)$ is the long-run fraction of jobs that observe n customers in the system at the moment an arbitrary job arrives. It is natural to ask whether $\pi(n)$ and $p(n)$, as defined by (2.4.2), 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.

Since $A(t) \to \infty$ as $t \to \infty$, it is reasonable that (see 2.7.4 for a proof)

$$\begin{aligned} \pi(n) &= \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=n} = \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{\infty} \mathbb{1}_{A_k\le t, L(A_k-)=n} \\ &= \lim_{t\to\infty} \frac{A(n,t)}{A(t)}, \end{aligned} \tag{2.7.2}$$

where we use (2.4.1a) in the last row. But, with (2.1.1),

$$\frac{A(n,t)}{t} = \frac{A(t)}{t}\frac{A(n,t)}{A(t)} \to \lambda\pi(n), \quad \text{as } t \to \infty, \tag{2.7.3}$$

while by (2.4.4),

$$\frac{A(n,t)}{t} = \frac{A(n,t)}{Y(n,t)}\frac{Y(n,t)}{t} \to \lambda(n)p(n), \quad \text{as } t \to \infty.$$

Thus

$$\lambda\pi(n) = \lambda(n)p(n). \tag{2.7.4}$$

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*. Thus, for the $M/M/1$ queue in particular,

$$\pi(n) = p(n) = (1-\rho)\rho^n.$$

2.7.1. *[Companion 2.7.1] Show for the case of 2.4.2 that $\pi(0) = 1$ and $\pi(n) = 0$, for $n > 0$.*

2.7.2. *[Companion 2.7.2] Check that (2.7.4) holds for the system of 2.7.1.*

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 (2.7.2),

$$\delta(n) = \lim_{t\to\infty} \frac{D(n,t)}{D(t)} \tag{2.7.5}$$

as the long-run fraction of jobs that leave n jobs *behind*. From (2.4.3)

$$\frac{A(t)}{t}\frac{A(n,t)}{A(t)} = \frac{A(n,t)}{t} \approx \frac{D(n,t)}{t} = \frac{D(t)}{t}\frac{D(n,t)}{D(t)}.$$

Taking limits at the left and right, and using (2.1.3), we obtain for (queueing) systems in which customers arrive and leave as single units that

$$\lambda\pi(n) = \delta\delta(n). \tag{2.7.6}$$

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.,

$$\lambda = \delta \iff \pi(n) = \delta(n). \tag{2.7.7}$$

Note that the derivation of this fact does not depend on the distribution of the inter-arrival or service times. Hence, it is true for the rate-stable $G/G/1$ queue.

2.7.3. *[Companion 2.7.3] When $\lambda \neq \delta$, is $\pi(n) \geq \delta(n)$?*

2.7.4. *[Companion 2.7.6] There is a subtle problem in the transition from* (2.7.1) *to* (2.7.2) *and the derivation of* (2.7.3)*: $\pi(n)$ 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* (2.7.2) *is valid.*

With the PASTA property we can determine the distribution of the inter-departure times of the $M/M/1$ queue. Observing that in a network of queues the departures from one queueing station form the arrivals at another station, we can use this result to analyze networks of queues

2.7.5. *[Companion 2.7.7] Try to prove* Burke's law *which states that the departure process of the $M/M/1$ queue is a Poisson process with rate λ.*

2.8 LITTLE'S LAW

There is an important relation between the average time $\mathsf{E}[W]$ a job spends in the system and the long-run time-average number $\mathsf{E}[L]$ of jobs that is contained in the system, which is called *Little's law*:

$$\mathsf{E}[L] = \lambda \mathsf{E}[W]. \tag{2.8.1}$$

2.8.1 provides a proof of this under some simple conditions. In the forthcoming sections, we will apply Little's law often. Part of the usefulness of Little's law is that it applies to all input-output systems, whether it is a queueing system or an inventory system or some much more general system.

We start by defining a few intuitively useful concepts. From (1.5.8), we see that

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \frac{1}{t}\int_0^t (A(s) - D(s))\,\mathrm{d}s$$

is the time-average of the number of jobs in the system during $[0,t]$. Next, the waiting time of the kth job is the time between the moment the job arrives and departs, that is,

$$W_k = \int_0^\infty \mathbb{1}_{A_k \le s < D_k}\,\mathrm{d}s.$$

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 $k \le A(T)$,

$$W_k = \int_0^T \mathbb{1}_{A_k \le s < D_k}\,\mathrm{d}s,$$

and for $s \le T$,

$$L(s) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le s < D_k} = \sum_{k=1}^{A(T)} \mathbb{1}_{A_k \le s < D_k}.$$

2.8.1. *[Companion 2.8.2] Prove Little's law under the assumptions that $A(T_i) = D(T_i)$ for an infinite number of times $\{T_i\}$ such $T_i \to \infty$ and that all limits exist.*

2.8.2. *[Companion 2.8.8] For a given single-server queueing system the average number of customers in the system is $\mathsf{E}[L] = 10$, 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 simple expressions for the average queue length and waiting times for the $M/M/1$ queue. The average waiting time $\mathsf{E}[W]$ in the entire system is the expected time in queue plus the expected time in service, i.e.,

$$\mathsf{E}[W] = \mathsf{E}\left[W_Q\right] + \mathsf{E}[S]. \tag{2.8.2}$$

By the PASTA property we have for the $M/M/1$ queue that

$$\mathsf{E}\left[W_Q\right] = \mathsf{E}[L]\,\mathsf{E}[S]. \tag{2.8.3}$$

2.8.3. *[Companion 2.8.9] Use Little's law to show for the $M/M/1$ queue that*

$$\mathsf{E}[W] = \frac{\mathsf{E}[S]}{1-\rho}, \qquad \mathsf{E}[L] = \frac{\rho}{1-\rho},$$
$$\mathsf{E}\left[L_Q\right] = \frac{\rho^2}{1-\rho}, \qquad \mathsf{E}[L_s] = \rho.$$

2.8.4. *[Companion 2.8.10] Why is* (2.8.3) not *true in general for the $M/G/1$ queue?*

The following problems show how combining PASTA with Little's law allows the analysis of some non-trivial practical queueing situations.

2.8.5. *[Hall 5.10][Companion 2.8.18] 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 $\mathsf{E}[W]$ for $c = 1,2,3,$ or 4 servers? How many employees should work in the tool crib?*

2.8.6. *[Hall 5.22][Companion 2.8.19] 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?*

2.8.7. *[Continuation of 2.8.6][Companion 2.8.21] Did you have to use the PASTA property to solve 2.8.6? If so, how did you use it? If not, why not?*

2.8.8. *[Continuation of 2.8.6][Companion 2.8.22] Suppose cabs can contain at most 4 riders, and the size of a party (i.e., a batch) has distribution B_k with $\mathsf{P}(B_k = i) = 1/7$ for $i = 1,\ldots,7$. 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...)*

In the above proof of Little's law we assumed that there is a sequence of moments $\{T_k, k = 0,1,\ldots\}$ at which the system is empty and such that $T_k < T_{k+1}$ and $T_k \to \infty$. 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.

2.8.9. *[Companion 2.8.23] Motivate in words (or with a derivation, if you prefer this) why the following is true:*

$$\sum_{k=1}^{A(t)} W_k \geq \int_0^t L(s)\,ds \geq \sum_{k=1}^{D(t)} W_k.$$

2.8.10. *[Companion 2.8.24] Take suitable limits (and assume all these limits exist) to show that*

$$\lambda\,\mathsf{E}[W] \geq \mathsf{E}[L] \geq \delta\,\mathsf{E}[W].$$

Make explicit all the points where you use the strong law of large numbers.

2.8.11. *[Companion 2.8.25] 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.*

2.8.12. *[Companion 2.8.26] 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.)*

2.9 $M^X/M/1$ 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.

Assume that jobs arrive as a Poisson process with rate λ 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 kth 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 $\mathsf{P}(B = k) = f(k)$ be given. The service time of each item is $\mathsf{E}[S]$.

2.9.1. *Explain that the average time to serve an entire batch is $\mathsf{E}[B]\,\mathsf{E}[S]$, so that the load must given by $\rho = \lambda\,\mathsf{E}[B]\,\mathsf{E}[S]$.*

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:

$$\mathsf{E}\left[W_Q\right] = \frac{1+C_s^2}{2}\,\frac{\rho}{1-\rho}\,\mathsf{E}[B]\,\mathsf{E}[S] + \frac{1}{2}\,\frac{\rho}{1-\rho}\,\mathsf{E}[S], \tag{2.9.1}$$

where C_s^2 is the SCV of the batch size distribution. By applying (2.8.3), it follows right away that the expected number of items in the system takes the form

$$\mathsf{E}[L] = \frac{\mathsf{E}\left[W_Q\right]}{\mathsf{E}[S]} = \frac{1+C_s^2}{2}\,\frac{\rho}{1-\rho}\,\mathsf{E}[B] + \frac{1}{2}\,\frac{\rho}{1-\rho}. \tag{2.9.2}$$

Note that $\rho < 1$ is required, as usual.

Before deriving the above, let us try to use it.

2.9.2. *[Companion 2.9.3] If the batch size is geometrically distributed with success probability p , what is $\mathsf{E}[L]$?*

2.9.3. *[Companion 2.9.4] 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*

$$\mathsf{P}(B=1) = \mathsf{P}(B=2) = \mathsf{P}(B=3) = \frac{1}{3}.$$

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 decreases if batch sizes were constant and equal to 2; hence the load is the same in both cases.

2.9.4. *[Companion 2.9.6] Show that $\mathsf{E}\left[W_Q(M^X/M/1)\right] \geq \mathsf{E}\left[W_Q(M/M/1)\right]$ 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 (2.9.1). 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 $\mathsf{E}\left[L_Q^B\right]$ for the number of batches in queue and $\mathsf{E}\left[L_S^B\right]$ 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

$$\mathsf{E}\left[W_Q\right] = \mathsf{E}[L]\,\mathsf{E}[S] = \left(\mathsf{E}\left[L_Q^B\right]\mathsf{E}[B] + \mathsf{E}\left[L_S^B\right]\right)\mathsf{E}[S].$$

We also see that the average time a batch spends in queue is

$$\mathsf{E}\left[W_Q^B\right] = \mathsf{E}\left[L_Q^B\right]\mathsf{E}[B]\,\mathsf{E}[S] + \mathsf{E}\left[L_S^B\right]\mathsf{E}[S].$$

Hence, $\mathsf{E}\left[W_Q\right] = \mathsf{E}\left[W_Q^B\right]$.

2.9.5. *[Companion 2.9.7] Use Little's law to show that*

$$\mathsf{E}\left[W_Q^B\right] = \frac{\mathsf{E}[L_S]}{1-\rho}\,\mathsf{E}[S].$$

Clearly, we are done if we can find an expression for $\mathsf{E}[L_S]$. For this we can use the renewal reward theorem; in fact, we can use 2.2.1 as inspiration. (Solve this exercise if you have not done yet.) Define $Y(t) = \int_0^t L_S^B(s)\mathrm{d}s$ to see that $\mathsf{E}\left[L_S^B\right] = Y = \lim_{t\to\infty} Y(t)/t$.

2.9.6. *Use the renewal reward theorem with the sampling epochs $T_k = D_k$ to prove that*

$$\mathsf{E}\left[L_S^B\right] = \lambda\frac{\mathsf{E}\left[B^2\right]}{2}\mathsf{E}[S] + \frac{\rho}{2}.$$

2.9.7. *[Companion 2.9.11] Use 2.9.6 to conclude* (2.9.1).

2.10 $M/G/1$ 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.

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

$$\mathsf{E}\left[W_Q\right] = \frac{1+C_s^2}{2}\frac{\rho}{1-\rho}\mathsf{E}[S]. \tag{2.10.1}$$

Before deriving this formula, let us apply it.

2.10.1. *[Companion 2.10.4] 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* $\mathsf{E}\left[L_Q\right]$, $\mathsf{E}[L]$, $\mathsf{E}\left[W_Q\right]$, $\mathsf{E}[W]$.

2.10.2. *[Companion 2.10.5] 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* $\mathsf{E}[S] = 2$ *hours,* $C_s^2 = 1/2$*, and the shop is open for 8 hours. What is* $\mathsf{E}\left[W_Q\right]$*?*

Suppose the expected waiting time has to be reduced to 1h. How to achieve this?

2.10.3. *[Hall 5.16] [Companion 2.10.6] 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* $\mathsf{E}\left[L_Q\right]$*? Which would minimize* $\mathsf{E}[L]$*?*

To derive the PK-formula, suppose at first that we know the expected *remaining service time* $\mathsf{E}[S_r]$, i.e., the expected time it takes to complete the job in service, if present, at the time a job arrives.

2.10.4. *[Companion 2.10.12] Show that, given* $\mathsf{E}[S_r]$*,*

$$\mathsf{E}\left[W_Q\right] = \frac{\mathsf{E}[S_r]}{1-\rho}. \tag{2.10.2}$$

It remains to compute the average remaining service time $\mathsf{E}[S_r]$ for generally distributed service times. Just like in Section 2.9 we use the renewal reward theorem. Consider the kth 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 $D_k = \tilde{A}_k + S_k$.

2.10.5. *[Companion 2.10.13] Use Fig. 13 to explain that the remaining service time of job* k *at time* s *is given by* $(D_k - s)\mathbb{1}_{\tilde{A}_k \le s < D_k}$*. 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*.*

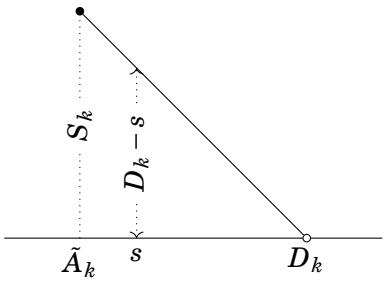


Figure 13: Remaining service time.

2.10.6. *[Companion 2.10.14] Apply the renewal reward theorem to the result of 2.10.5 to prove that*

$$\mathsf{E}[S_r] = \frac{\lambda}{2}\mathsf{E}\left[S^2\right]. \tag{2.10.3}$$

Then simplify to get (2.10.1).

2.10.7. *[Companion 2.10.18] Show from* (2.10.3) *that*

$$\mathsf{E}[S_r \,|\, S_r > 0] = \frac{\mathsf{E}\left[S^2\right]}{2\mathsf{E}[S]}. \tag{2.10.4}$$

2.11 $M^X/M/1$ QUEUE LENGTH DISTRIBUTION

In Sections 2.9 and 2.10 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, for instance, the loss probability $\mathsf{P}(L > n)$, we need expressions for the stationary distribution $\pi(n) = \mathsf{P}(L = n)$ of the number of jobs in the system. Here we present a numerical, recursive, scheme to compute these probabilities.

To find $\pi(n)$, $n = 0, 1, \ldots$, we turn again to level-crossing arguments. However, the reasoning that led to the level-crossing equation (2.4.3) 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 kth 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. 14 shows all up- and down-crossings of some level n. The down-crossing rate is easy: just as in Fig. 9 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 $m \in \{0, 1, \ldots, n-1\}$. 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 $n - m$.

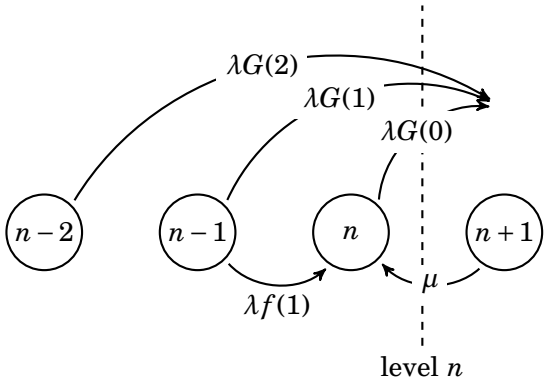


Figure 14: Level crossing of level n. Observe that when the system is in state $n-2$, the arrival of any batch larger than 2 ensures that level n is crossed from below. The rate at which such events happen is $\lambda\pi(n-2)G(2)$. Similarly, in state $n-1$, the arrival of any batch larger than one item ensures that level n is crossed, and this occurs with rate $\lambda\pi(n-1)G(1)$, and so on.

2.11.1. *[Companion 2.11.4] Assuming that the limits exist, show that*

$$\lim_{t\to\infty} \frac{A(m,n,t)}{t} = \lambda\pi(m)G(n-m).$$

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 $t\to\infty$, and using 2.11.1 results in the level-crossing equation for the $M^X/M/1$ queue:

$$\lambda \sum_{m=0}^{n} \pi(m)G(n-m) = \mu\pi(n+1). \tag{2.11.1}$$

2.11.2. *[Companion 2.11.5] Provide an interpretation of* (2.11.1) *in terms of a thinned Poisson arrival process.*

It is left to find the normalization constant. As the recursion (2.11.1) does not lead to a closed form expression for $\pi(n)$, such as (2.5.1), 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^2 is some large number. 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, λ, and μ on the system's performance.

2.11.3. *[Companion 2.11.7] Why is* (2.7.7)*, i.e., $\pi(n) = \delta(n)$, not true for the $M^X/M/1$ batch queue? Provide an example.*

2.11.4. *[Companion 2.11.10] Substitute recursion* (2.11.1) *for $\pi(n)$ into the expression $\mathsf{E}[L] = \sum_{n=0}^{\infty} n\pi(n)$ and derive* (2.9.2)*.*

2.11.5. *[Companion 2.11.11] Implement the recursion* (2.11.1) *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

[2] An interesting question, why should it decrease monotonically after some, large, N?

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.

2.11.6. *[Companion 2.11.12] Derive a set of recursions, analogous to* (2.11.1)*, to compute* $\pi(n)$ *for the* $M^X/M/1/K$ *queue with complete rejection.*

2.12 $M/G/1$ QUEUE LENGTH DISTRIBUTION

In Section 2.11 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 $p(n) = \mathsf{P}(L = n)$ 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.

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 (2.7.7) 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 kth 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 $f(j) = \mathsf{P}(Y = j)$; write $G(j) = \mathsf{P}(Y_k > j)$ for the survivor function.

2.12.1. *[Companion 2.12.2] Explain that*

$$\mathsf{P}(Y_k = j) = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!}\, dF(x), \tag{2.12.1}$$

where F *is the distribution of the service times.*

2.12.2. *[Companion 2.12.5] If* $S \sim \mathrm{Exp}(\mu)$*, show that*

$$G(j) = \sum_{k=j+1}^{\infty} f(k) = \left(\frac{\lambda}{\lambda+\mu}\right)^{j+1}. \tag{2.12.2}$$

2.12.3. *[Companion 2.12.6] Design a suitable numerical method to evaluate* (2.12.1) *for more general distribution functions* F*.*

Let us concentrate on a down-crossing of level n, see Fig. 15; 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,

$$\text{Down-crossing of level } n \iff \mathbb{1}_{L(D_{k-1})=n+1}\, \mathbb{1}_{L(D_k)=n} = 1.$$

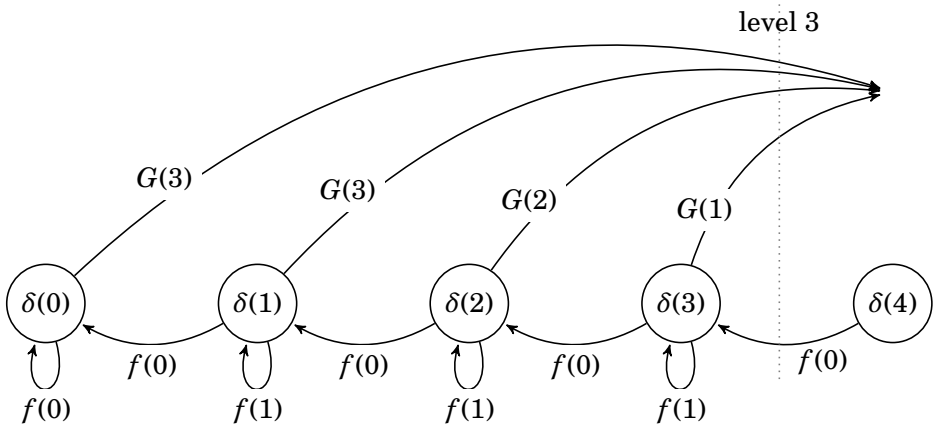


Figure 15: Level 3 is crossed from below with rate $\delta\delta(0)G(3)+\delta\delta(1)G(3)+\cdots\delta\delta(3)G(1)$ and crossed from above with rate $\delta\delta(4)f(0)$.

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

$$D(n+1,0,t)=\sum_{k=1}^{D(t)}\mathbb{1}_{L(D_{k-1})=n+1}\,\mathbb{1}_{Y_k=0}.$$

2.12.4. *[Companion 2.12.7] Show that*

$$\lim_{t\to\infty}\frac{D(n+1,0,t)}{t}=\delta\delta(n+1)f(0),$$

where $f(0)=\mathsf{P}(Y=0)$.

Before we deal with the up-crossing, it is important to do the next exercise.

2.12.5. *[Companion 2.12.8] 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* $L(D_{k-1})=0$?

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),

$$\mathbb{1}_{L(D_k)>n} = \mathbb{1}_{L(D_{k-1})=0}\,\mathbb{1}_{Y_k>n} \implies \text{Up-crossing of level } n.$$

Again we define proper counting functions, divide by t, and take suitable limits to find for the up-crossing rate

$$\delta\delta(0)G(n) + \delta \sum_{m=1}^{n} \delta(m)G(n-m+1). \tag{2.12.3}$$

Equating the down-crossing and up-crossing rates and dividing by δ 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., (2.7.7), hence in particular for the $M/G/1$ queue length process, we arrive at

$$f(0)\pi(n+1) = \pi(0)G(n) + \sum_{m=1}^{n} \pi(m)G(n+1-m). \tag{2.12.4}$$

Clearly, we have again obtained a recursion with which we can compute the state probabilities.

2.12.6. *[Companion 2.12.9] Provide the details behind the derivation of* (2.12.3).

2.12.7. *[Companion 2.12.10] 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* (2.12.4).

2.13 GRAPHICAL SUMMARIES

We finish this chapter with providing two summaries in graphical form to clarify how all concepts developed in this chapter relate.

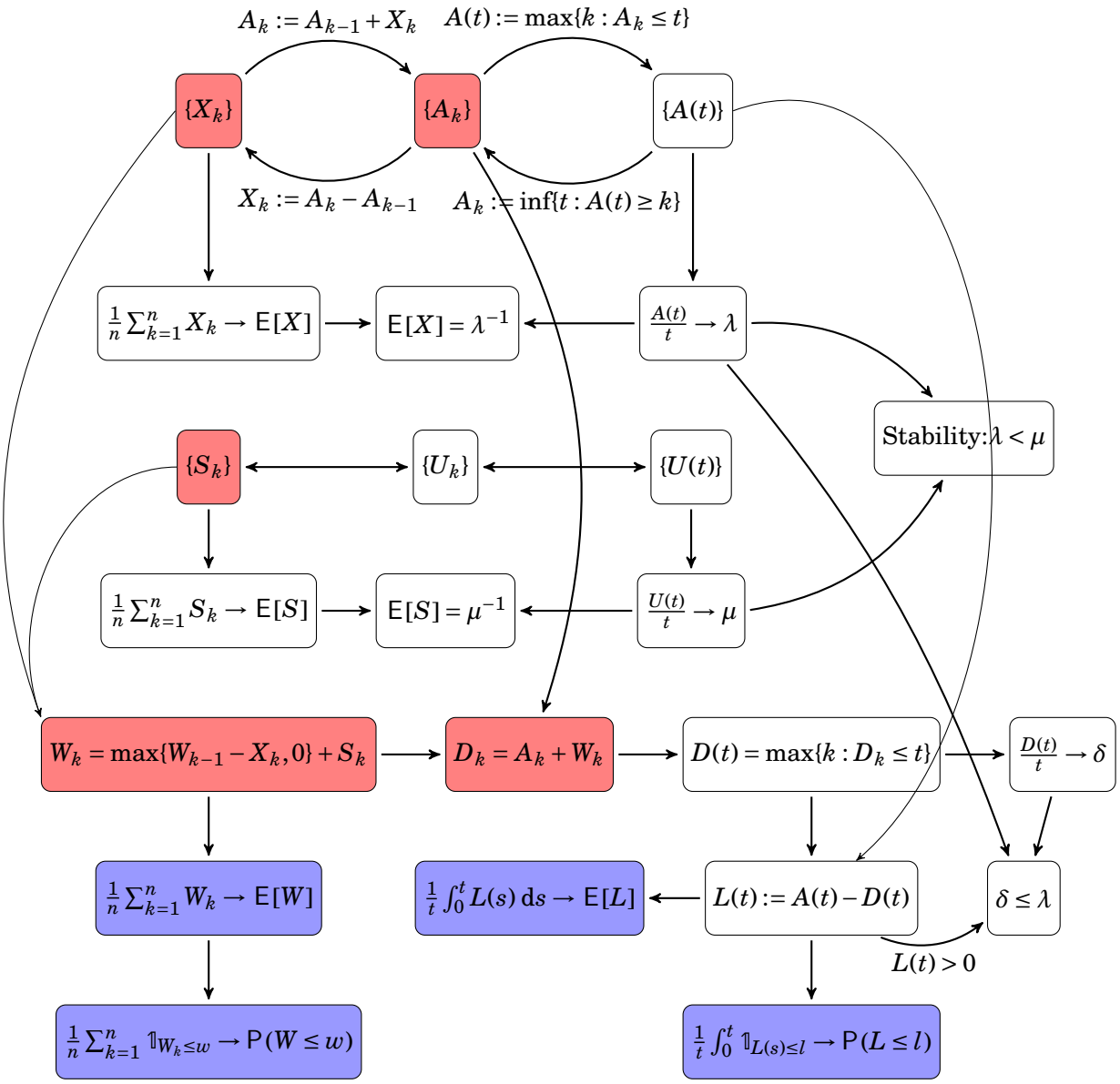


Figure 16: Here we sketch the relations between the construction of the $G/G/1$ queue from the primary data, i.e., the inter-arrival times $\{X_k; k \ge 0\}$ and the service times $\{S_k; k \ge 0\}$, and different performance measures.

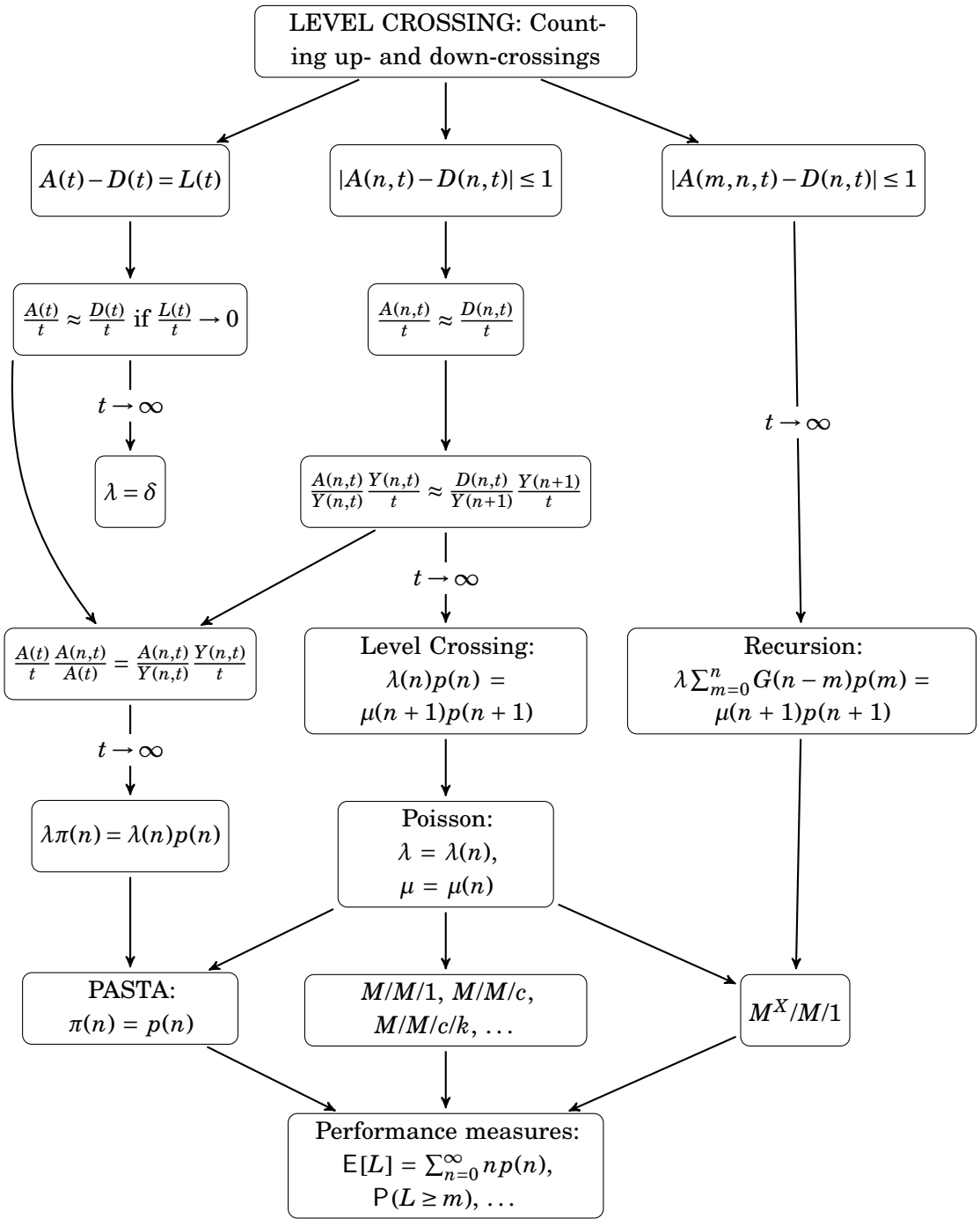


Figure 17: With level-crossing arguments we can derive a number of useful relations. This figure presents an overview of these relations that we derive in this and the next sections.

3

APPROXIMATE MODELS

In this chapter we first consider the very useful formula of Sakasegawa to approximate the average waiting time in queue for the $G/G/c$ queue. We then illustrate how to use this formula to estimate waiting time in three examples in which the service process is interrupted. In the first case 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 has 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, 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, quality problems or break downs can occur during a job's service. For each case we develop a model to analyze the influence of the interruptions on average job sojourn times.

3.1 $G/G/c$ QUEUE: APPROXIMATIONS

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 SCV of the arrivals at the second station $C_{a,2}^2$ is most probably smaller than 1. 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. Thus, the $M/G/1$ waiting time formula should not be naively applied to approximate the average waiting time of the $G/G/1$ queue.

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

$$\mathsf{E}\left[W_Q\right] = \frac{C_a^2 + C_s^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} \mathsf{E}[S], \tag{3.1.1}$$

where

$$\rho = \frac{\lambda \mathsf{E}[S]}{c}$$

is the load of the station (but not of the individual machines) and $C_a^2 = \mathsf{V}[X]/(\mathsf{E}[X])^2$ is the SCV of the inter-arrival times. This formula is reasonably accurate; for related expressions we refer to Bolch et al. [2006] and Hall [1991].

It is crucial to memorize the *scaling* relations that can be obtained from the $G/G/c$ waiting time 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 capacity, variability and service times. Roughly:

1. $\mathsf{E}\left[W_Q\right] \sim (1-\rho)^{-1}$. The consequence is that the waiting time increases *very steeply* when ρ is large. Hence, the waiting time is very sensitive to the actual value of ρ when ρ is large.

2. $\mathsf{E}[W_Q] \sim C_a^2$ and $\mathsf{E}[W_Q] \sim C_s^2$. Hence, reductions of the variation of the inter-arrival and service times do affect the waiting time, but only linearly.

3. $\mathsf{E}[W_Q] \sim \rho^{\sqrt{2c}}\,\mathsf{E}[S]/c$. This means that, from the perspective a job in queue, not only the average services are c times as short, the average waiting time decreases faster, due to the $\sqrt{c}$ effect in the power ρ (and $\rho < 1$).

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. *[Companion 3.1.1] Show that the approximation* (3.1.1) *reduces to the result known for the M/M/*1 *and M/G/*1 *queues.*

3.1.2. *[Companion 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 λ be the arrival rate, μ the service rate, β the long-run fraction of customers lost, and ρ the average number of busy/occupied servers. Show that*

$$\beta = 1 - \rho\frac{\mu}{\lambda}.$$

3.1.3. *[Companion 3.1.5] Consider the same single-server system as in* **??***, but now the customer service time is stochastic: with probability* 1/2 *a customer requires* 1 *minute and* 20 *seconds of service, and with probability* 1/2 *the customer requires only* 20 *seconds of service. What are ρ, C_a^2, and C_S^2?*

It is crucial to remember from the above exercises that knowledge of the utilization is not sufficient to characterize the average queue length.

3.1.4. *[Companion 3.1.7] (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.*

*As a first case, assuming a single server, estimate $\mathsf{E}[W_Q]$ 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 your previous answer is smaller.*

Clearly, Sakasegawa's equation requires an estimate of the SCV C_a^2 of the inter-arrival times and the SCV C_s^2 of the service times. 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, mostly just the date or the hour, perhaps, 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 day has been logged for some time so that we know $\{a_n, n = 1,\ldots,N\}$ 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

$$\tilde{\lambda} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} a_i, \quad \tilde{\sigma}^2 = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} a_i^2 - \tilde{\lambda}^2.$$

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 2.1. 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 $\{X_k, k = 1,2,\ldots\}$ with $X_k = A_k - A_{k-1}$, are i.i.d. with mean $1/\lambda$ and standard deviation σ. (Observe that σ is not the same as $\tilde{\sigma}$ above.) Note that C_a^2 is defined in terms of λ and σ as:

$$C_a^2 = \frac{\mathsf{V}[X_i]}{(\mathsf{E}[X_i])^2} = \frac{\sigma^2}{1/\lambda^2} = \lambda^2\sigma^2.$$

Next, let A_k be the arrival time of the kth 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):

$$\mathsf{P}(A(t) < k) = \mathsf{P}(A_k > t).$$

Since the inter-arrival times have finite mean and second moment by assumption, we can apply the central limit law to obtain that, as $k \to \infty$,

$$\frac{A_k - k/\lambda}{\sigma\sqrt{k}} \to N(0,1),$$

where $N(0,1)$ is a standard normal random variable with distribution $\Phi(\cdot)$. Similarly,

$$\frac{A(t) - \lambda t}{\alpha\sqrt{t}} \to N(0,1)$$

for an α that is yet to be determined. Thus, $\mathsf{E}[A(t)] = \lambda t$ and $\mathsf{V}[A(t)] = \alpha^2 t$.

Using that $\mathsf{P}(N(0,1) \leq y) = \mathsf{P}(N(0,1) > -y)$ (and $\mathsf{P}(N(0,1) = y) = 0$) we have that

$$\begin{aligned}
\Phi(y) &\approx \mathsf{P}\left(\frac{A_k - k/\lambda}{\sigma\sqrt{k}} \leq y\right) \\
&= \mathsf{P}\left(\frac{A_k - k/\lambda}{\sigma\sqrt{k}} > -y\right) \\
&= \mathsf{P}\left(A_k > \frac{k}{\lambda} - y\sigma\sqrt{k}\right).
\end{aligned}$$

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 $\mathsf{P}(A_k > t_k) = \mathsf{P}(A(t_k) < k)$. With this we get,

$$\begin{aligned}
\Phi(y) &\approx \mathsf{P}(A_k > t_k) \\
&= \mathsf{P}(A(t_k) < k) \\
&= \mathsf{P}\left(\frac{A(t_k) - \lambda t_k}{\alpha\sqrt{t_k}} < \frac{k - \lambda t_k}{\alpha\sqrt{t_k}}\right).
\end{aligned}$$

Since $(A(t_k) - \lambda t_k)/\alpha\sqrt{t_k} \to N(0,1)$ as $t_k \to \infty$, the above implies that

$$\frac{k - \lambda t_k}{\alpha\sqrt{t_k}} \to y,$$

as $t_k \to \infty$. 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 $t_k \to \infty$ is implied by (and implies) $k \to \infty$, we therefore want that α is such that

$$\frac{\lambda\sigma\sqrt{k}}{\alpha\sqrt{k/\lambda + \sigma\sqrt{k}}} y \to y,$$

as $k \to \infty$. This is precisely the case when

$$\alpha = \lambda^{3/2}\sigma.$$

Finally, for t large (or, by the same token k large),

$$\frac{\sigma_k^2}{\lambda_k} = \frac{\mathsf{V}[A(t)]}{\mathsf{E}[A(t)]} \approx \frac{\alpha^2 t}{\lambda t} = \frac{\alpha^2}{\lambda} = \frac{\lambda^3\sigma^2}{\lambda} = \lambda^2\sigma^2 = C_a^2,$$

where the last equation follows from the above definition of C_a^2. The proof is complete. □

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, inn 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 change from type or location, starts serve 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.

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 λ_r and λ_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 $\mathsf{E}[S]_0$, 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. *[Companion 3.2.1] Show that the average time needed to form (or assemble) a batch is given by*

$$\mathsf{E}[W_r] = \frac{B-1}{2\lambda_r}. \tag{3.2.1}$$

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* $\mathsf{E}[S]$ 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. *[Companion 3.2.2] Motivate that the effective processing time should be defined as*

$$\mathsf{E}[S] = \mathsf{E}[S_0] + \frac{\mathsf{E}[R]}{B}. \tag{3.2.2}$$

Observe then that the load becomes

$$\rho = \lambda\left(\mathsf{E}[S_0] + \frac{\mathsf{E}[R]}{B}\right).$$

3.2.3. *[Companion 3.2.3] Explain that the load can also be written as*

$$\rho = \lambda_B(B\,\mathsf{E}[S_0] + \mathsf{E}[R]),$$

where λ_B is the arrival rate of batches and $B\,\mathsf{E}[S_0] + \mathsf{E}[R]$ is the service time an entire batch.

3.2.4. *[Companion 3.2.4] Show that the requirement $\rho < 1$ leads to the following constraint on the minimal batch size B*

$$B > \frac{\lambda\,\mathsf{E}[R]}{1-\lambda\,\mathsf{E}[S_0]}.$$

Observe that we nearly have all elements to apply Sakasegawa's formula to the batch: the service time of a batch is $\mathsf{E}[R] + B\,\mathsf{E}[S_0]$ and the load ρ is given above. Therefore it remains to find $C^2_{a,B}$ and $C^2_{s,B}$, for the *batches, not the items*.

3.2.5. *[Companion 3.2.5] Explain that the SCV of the batch inter-arrival times is given by*

$$C^2_{a,B} = \frac{C^2_a}{B}. \tag{3.2.3}$$

3.2.6. *[Companion 3.2.6] Show that the SCV $C^2_{s,B}$ of the service times of the batches takes the form*

$$C^2_{s,B} = \frac{B\,\mathsf{V}[S_0] + \mathsf{V}[R]}{(B\,\mathsf{E}[S_0] + \mathsf{E}[R])^2}. \tag{3.2.4}$$

Observe that we now have all elements to fill in Sakasegawa's formula. Thus, 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 time a item spends at the server is $\mathsf{E}[R] + B\,\mathsf{E}[S_0]$.

3.2.7. *[Companion 3.2.7] Show that, when items can leave right after being served, the time at the server is given by*

$$\mathsf{E}[R] + \frac{B+1}{2}\,\mathsf{E}[S_0]. \tag{3.2.5}$$

3.2.8. *[Companion 3.2.8] What important insights can you learn from the above about setting proper batch sizes?*

3.2.9. *[Companion 3.2.9] 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.

3.3 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.

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 $i+2$.

With the model, we can obtain quantitative insights into the effects of reducing adjustment times or the variability of these adjustments times. For instance, we might decide to do fewer adjustments, so that p decreases, but at the expense of larger average outage times. Now we can analyze the consequences of such decisions without needing to actually do the experiments in real life.

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. They are given by

$$\mathsf{E}[S] = \mathsf{E}[S_0] + \frac{\mathsf{E}[R]}{B}, \tag{3.3.1}$$

$$\mathsf{V}[S] = \mathsf{V}[S_0] + \frac{\mathsf{V}[R]}{B} + (B-1)\left(\frac{\mathsf{E}[R]}{B}\right)^2. \tag{3.3.2}$$

Thus, the effective server load including down-times is $\rho = \lambda\,\mathsf{E}[S]$, and we can compute SCV of the effective job processing times. We have now all elements to fill in the $G/G/1$ waiting time formula!

3.3.1. *[Companion 3.3.1] A machine requires an adjustment with an average of 5 hours and a standard deviation of 2 hours. 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. Interruptions occur on average between 30 jobs. Compute the average waiting time in queue.*

3.3.2. *[Companion 3.3.2] Show that the average effective processing time is given by* (3.3.1).

3.3.3. *[Companion 3.3.4] Derive $V(S)$ in* (3.3.2).

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, to use the $G/G/1$ waiting time formula it suffices to find expressions for $\mathsf{E}[S]$ and $\mathsf{V}[S]$. Thus, this will be our task for the rest of the section. We remark in passing that the results and the derivation are of general interest.

Supposing that N interruptions occur during the net service time S_0 of a job, the effective service time will be

$$S = S_0 + \sum_{i=1}^{N} R_i.$$

A common assumption is that the time between two interruptions is $\text{Exp}(\lambda_f)$, hence is memoryless. Consequently, the number of interruptions N that occur during the net service time S_0 is Poisson distributed with mean $\mathsf{E}[N] = \lambda_f \mathsf{E}[S_0]$. 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. With this,

$$\mathsf{E}[S] = \frac{\mathsf{E}[S_0]}{A}. \tag{3.4.1}$$

Finally, by assuming that repair times are exponentially distributed with mean $\mathsf{E}[R]$, we can find that

$$C_s^2 = C_0^2 + 2A(1-A)\frac{\mathsf{E}[R]}{\mathsf{E}[S_0]}, \tag{3.4.2}$$

where C_0^2 is the SCV of S_0, i.e., the service time without interruptions. It is important to realize that

$$\rho = \lambda \mathsf{E}[S] = \lambda \frac{\mathsf{E}[S_0]}{A},$$

hence the server load increases due to failures.

Before deriving the above expressions, let us see how to apply them.

3.4.1. *[Companion 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. *[Companion 3.4.4] Show that* $\mathsf{E}[S] = \mathsf{E}[S_0] + \mathsf{E}[R]\,\mathsf{E}[N]$*; then show* (3.4.1).

3.4.3. *[Companion 3.4.9] The derivation of* C_s^2 *is a bit more involved. To see this, explain that* $\mathsf{V}[S] \neq \mathsf{V}[S_0] + \mathsf{V}\left[\sum_{i=0}^{N} R_i\right]$.

3.4.4. *[Companion 3.4.16] Show that*

$$C_s^2 = \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} = C_0^2 + \frac{\lambda_f \mathsf{E}\left[R^2\right] A^2}{\mathsf{E}[S_0]},$$

3.4.5. *[Companion 3.4.17] With the above assumption on the distribution of* R*, show that*

$$C_s^2 = C_0^2 + 2A(1-A)\frac{\mathsf{E}[R]}{\mathsf{E}[S_0]}.$$

4

QUEUEING NETWORKS

We refer to the relevant sections of Zijm's book for background. Here we just include the solutions and repair a few typos.

4.1 OPEN SINGLE-CLASS PRODUCT-FORM NETWORKS

Theory and Exercises

The remark above Zijm.Eq.2.11 is not entirely correct. Remove the sentence: 'These visit ratios satisfy ... up to a multiplicative constant'.

I don't like the derivation of Zijm.Eq.2.20. The appearance of the visit ratios λ_i/γ seems to come out of thin air. The argument should be like this. Consider the entire queueing network as one 'box' in which jobs enter at rate $\gamma = \sum_{i=1}^{M} \gamma_i$. Assuming that there is sufficient capacity at each station, i.e., $\lambda_i < c_i\mu_i$ at each station i, the output rate of the 'box' must also be γ. Thus, by applying Little's law to the 'box', we have that

$$\mathsf{E}[L] = \gamma\,\mathsf{E}[W].$$

It is also evident that the average total number of jobs must be equal to the sum of the average number of jobs at each station:

$$\mathsf{E}[L] = \sum_{i=1}^{M} \mathsf{E}[L_i].$$

Applying Little's law to each station separately we get that $\mathsf{E}[L_i] = \lambda_i\,\mathsf{E}[W_i]$. Filling this into the above,

$$\mathsf{E}[W] = \frac{\mathsf{E}[L]}{\gamma} = \sum_{i=1}^{M} \frac{\mathsf{E}[L_i]}{\gamma} = \sum_{i=1}^{M} \frac{\lambda_i\,\mathsf{E}[W_i]}{\gamma},$$

where we recognize the visit ratios.

4.1.1. *[Linear algebra refresher][Companion 4.1.1] Can you find an example to show for two matrices A and B that $AB \neq BA$, hence $xA \neq Ax$.*

4.1.2. *[Linear algebra refresher 2][Companion 4.1.2] Suppose the matrix A has an eigenvalue 0. What is the geometric meaning of this fact?*

4.1.3. *[Companion 4.1.3] Zijm.Ex.2.2.1*

4.1.4. *[Companion 4.1.4] Zijm.Ex.2.2.2*

4.1.5. *[Companion 4.1.5] Zijm.Ex.2.2.3*

4.1.6. *[Companion 4.1.6] Zijm.Ex.2.2.4*

4.1.7. *[Companion 4.1.7] Zijm.Ex.2.2.5. The problem is not entirely correctly formulated. It should be, if for at least one i, $\sum_{j=1}^{M} P_{ij} < 1$ ...*

4.1.8. *[Companion 4.1.8] Zijm.Ex.2.2.6*

4.1.9. *[Companion 4.1.9] Show that Zijm.Eq.2.13 and 2.14 can be written as*

$$f_i(n_i) = \frac{1}{G(i)} \frac{1}{\Pi_{k=1}^{n_i} \min\{k, c_i\}} \left(\frac{\lambda_i}{\mu_i}\right)^{n_i}.$$

4.1.10. *[Companion 4.1.10] We have a two-station single-server open network. Jobs enter the network at the first station with rate γ. A fraction α returns from station 1 to itself; the rest moves to station 2. At station 2 a fraction β_2 returns to station 2 again, a fraction β_1 goes to station 1. Compute λ. What happens if $\alpha \to 1$ or $\beta_1 \to 0$?*

4.1.11. *[Companion 4.1.11] Zijm.Ex.2.2.8*

4.2 TANDEM QUEUES

Theory and Exercises

Consider two $M/M/1$ stations in tandem. Suppose we can remove the variability in the service processing times at one, but not both, of the servers. Which one is the better one to spend it on, in terms of reducing waiting times? After we obtained some insights into this question, we will provide a model to approximate the waiting time in a tandem of $G/G/1$ queues.

4.2.1. *[Companion 4.2.1] Assuming that jobs arrive at the first station at rate λ, and are served at rate μ_i at station i, show that the average queueing time for the tandem of two $M/M/1$ queues is given by*

$$\mathsf{E}\left[W_Q\right] = \frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}, \tag{4.2.1}$$

where $\rho_i = \lambda/\mu_i$ and $\mathsf{E}[S_i] = 1/\mu_i$, for $i = 1,2$.

4.2.2. *[Companion 4.2.2] Suppose we can remove all variability of the2 service process at the second station. Show that in this case the total time in queue is equal to*

$$\mathsf{E}\left[W_Q\right] = \frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{1}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}.$$

4.2.3. *[Companion 4.2.3] Suppose now that we reduce the variability of the service process of the first station. Motivate that*

$$\mathsf{E}\left[W_Q\right] = \frac{1}{2}\frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{1}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}$$

is a reasonable approximation of the queueing time. Compare this to the queueing time of the reference situation.

4.2.4. *[Companion 4.2.4] What do you conclude from the above exercises?*

For a tandem network of $G/G/1$ queues, observe that the SCV of the departure process $C_{d,i}^2$ of the ith station is the SCV of the arrival process $C_{a,i+1}^2$ at station $i+1$. Thus, if we have $C_{d,i}^2$ we can compute the average waiting time at station $i+1$ by means of the $G/G/1$ waiting time approximation.

To obtain an estimate for $C_{d,i}^2$ we reason as follows. Suppose that the load ρ_i at station i is very high. Then the server will seldom be idle, so that the departure process must be reasonably well approximated by the service process. If, however, the load is small, the server will be idle most of the time, and inter-departure times must be approximately distributed as the inter-arrival times. Based on this, we interpolate between these two extremes to get the approximation

$$C_{d,i}^2 \approx (1-\rho_i^2)C_{a,i}^2 + \rho_i^2 C_{s,i}^2. \tag{4.2.2}$$

4.2.5. *[Companion 4.2.5] What is C_d^2 for the $D/D/1$ queue according to (4.2.2)?*

4.2.6. *[Companion 4.2.6] What is C_d^2 for the $M/M/1$ queue according to (4.2.2)?*

4.2.7. *[Companion 4.2.7] Use (4.2.2) to show for the $G/D/1$ that $C_d^2 < C_a^2$.*

4.2.8. *[Companion 4.2.8] 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 $\mathsf{E}[S_1] = 20$ minutes and $\mathsf{E}[S_2] = 25$ minutes. What is the total time jobs spend on average in the system? What is the average number of jobs in the network?*

For a $G/G/c$ queue, we can use the following approximation

$$C_{d,i}^2 = 1 + (1-\rho_i^2)(C_{a,i}^2 - 1) + \frac{\rho_i^2}{\sqrt{c_i}}(C_{s,i}^2 - 1). \tag{4.2.3}$$

4.2.9. *[Companion 4.2.9] Show that (4.2.3) reduces to (4.2.2) for the $G/G/1$ queue.*

For the interested reader we refer to Zijm, Section 2.4.2, for a discussion of an extension for $G/G/c$ queues in tandem, and to networks. In particular, in networks we need to be concerned with output streams merging into a single input stream at one station, and the splitting of the output stream of a station to several other stations. The algorithm discussed in Zijm, Section 2.4.2, is mainly useful for numerical analysis. We will not discuss it here.

4.3 GORDON-NEWELL NETWORKS

4.3.1. *[Companion 4.3.1] Provide an interpretation of a single-server queueing server with a finite calling population in terms of a closed network.*

The formula with the visit ratios should be like this:

$$V_k = \sum_{j=0}^{M} V_j P_{jk},$$

i.e., the sum should start at index 0. This is to include the load/unload station.

Also, assume that the load/unload station has just one server.

You should realize that the algorithms discussed in this section are meant to be carried out by computers. Thus the results will be numerical, not in terms of formulas.

Mind the order of V and P in the computation of the visit ratios: do not mix up $VP = V$ with $PV = V$, as in general, $VP \neq PV$. We use $VP = V$.

4.3.2. *[Companion 4.3.2] Compute the visit ratios for a network with three stations such that all jobs from station 0 move to station 1, from station 1 all move to station 2, and from station 2 half of the jobs move to station 0 and the other half to station 1.*

4.3.3. *[Companion 4.3.3] Zijm.Ex.3.1.1*

4.3.4. *[Companion 4.3.4] Zijm.Ex.3.1.2*

4.3.5. *[Companion 4.3.5] Zijm.Ex.3.1.3*

4.3.6. *[Companion 4.3.6] Relate Zijm.Eq.3.3 to the form of the steady-state distribution of the number of jobs in an $M/M/c$ queue.*

4.3.7. *[Companion 4.3.7] Zijm.Ex.3.1.4*

4.3.8. *[Companion 4.3.8] Zijm.Ex.3.1.5*

4.4 MVA ALGORITHM

4.4.1. *[Companion 4.4.1] Consider two stations in tandem, stations 0 and 1. The service times are $\mathsf{E}[S_0] = 2 = 1/\mu_0$ and $\mathsf{E}[S_1] = 3 = 1/\mu_1$ hours. The routing matrix is*

$$P = \begin{pmatrix} 0 & 1 \\ 1/2 & 1/2 \end{pmatrix}.$$

Apply the MVA algorithm to this case.

4.4.2. *[Companion 4.4.2] Zijm.Ex.3.1.13. Assume that all stations have just one server.*

4.4.3. *[Companion 4.4.3] Zijm.Ex.3.1.14*

4.4.4. *[Companion 4.4.4] Implement the MVA algorithm in your preferred computer language and make Figure 3.2.*

5

QUEUEING CONTROL

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, a so-called N-policy can be used, which works as follows, cf. 1.3.5. As soon as the system becomes empty (and the server idle), we switch off the server. Then we wait until N or more jobs arrived and then switch on the server. The server processes jobs until the system is empty again, and then switches off again, 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 occupancy can also increase. In fact, this sometimes seems to the policy at dentists or hospitals: wait until the waiting room is quite full, and then start serving patients. Like this the server minimizes idle time, and, in the example of a GP, the server (GP) does not have to wait for patients that might be late.

In this chapter we study the $M/M/1$ queue and $M/G/1$ under a N-policy, and then consider a T-policy. We next focus on a policy that controls the server rate as a function of waiting time in the system.

We point out that the techniques developed in this chapter extend (way) beyond just queueing theory; they are worth memorizing. Moreover, we illustrate many tools and results of the previous chapters.

5.1 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 switched. 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 $\mathsf{E}[L]$ of jobs in the system; the first resulted from the analysis of the $M/M/1$ queue in Section 2.5, the second from Little's law, cf. 2.8.3

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 1.4.8 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 1.4.7 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:

$$T(q) = \alpha T(q+1) + \beta T(q-1) + \frac{1}{\lambda+\mu}. \tag{5.1.1}$$

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.

5.1.1. *Provide an intuitive explanation for this formula, and show that* $T(q) = q/(\mu-\lambda)$ *solves* (5.1.1).

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

$$V(q) = \alpha V(q+1) + \beta V(q-1) + h\frac{q}{\lambda+\mu}. \tag{5.1.2}$$

To solve this, observe that in (5.1.1) 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 which leads us to the guess aq^2+bq+c for $V(q)$. Thus, we substitute this into the above expression and then try to solve for a, b and c.[1] As $V(0) = 0$, it follows already that $c = 0$. Thus, it remains to find a and b.

5.1.2. *Use the ideas of 5.1.1 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 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.

5.1.3. *Explain that* $C(1) = 1/\lambda + T(1)$ *for the* $M/M/1$ *queue.*

5.1.4. *Use the renewal-reward theorem to explain this the following relation:*

$$\frac{V(1)}{C(1)} = \frac{V(1)}{1/\lambda + T(1)} = h\,\mathsf{E}[L],$$

where $\mathsf{E}[L]$ *is given by* (2.5.2). *Then use the above expressions for* $V(1)$ *and* $C(1)$ *to verify this.*

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 1.4.4, the expected time to switch on must be equal to N/λ.

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 qth job (the job that sees $q-1$ jobs in the system). Then, by the next exercise,

$$W(q) = W(q-1) + h\frac{q-1}{\lambda} = h\frac{q(q-1)}{2\lambda}.$$

5.1.5. *Explain the above recursion, and derive the right-hand side.*

[1] 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.

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 $C(N) = N/\lambda + T(N)$. (We choose not to specify this in terms of N, as not much insight will be gained from this.)

Finding the optimal N is easy (from a practical point): just plot the time-average cost a function of N, and identify the minimum. (It is easy to turn the above procedure into a computer program.) It is possible to find more efficient algorithms, but we do not discuss this here. Besides, and again from a practical point of view, it seems unlikely to use an N-policy when we expect that N is a very large number. Typically, the optimal N will less than 100, or so; hence, we just need to compute 100 or so cost functions.

BIBLIOGRAPHY

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.

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.

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.

a_k = Number of arrivals in the kth period

$A(t)$ = Number of arrivals in $[0,t]$

A_k = Arrival time of kth job

$\tilde{A}_k$ = Start of service of kth job

c_n = Service/production capacity in the nth period

d_n = Number of departures in the nth 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 kth 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

$\mathsf{E}[L]$ = Long run (time) average of the number of jobs in the system

$\mathsf{E}[Q]$ = Long run (time) average of the number of jobs in queue

$\mathsf{E}[L_S]$ = 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 kth job, or at the *end* of the kth period

S_k = Service time required by the kth job

$S(t)$ = Total service time available in $[0,t]$

S = Generic service time of a job

t = Time

W_k = Time in the system of kth job

$W_{Q,k}$ = Time in the queue of kth job

$\mathsf{E}[W]$ = Sample average of the sojourn time

$\mathsf{E}\left[W_Q\right]$ = 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

δ = Departure rate

λ = Arrival rate

μ = Service rate

$\pi(n)$ = Stationary probability that an arrival sees n jobs in the system

ρ = Load on the system

$$\rho = \lambda \frac{\mathsf{E}[S]}{c}$$

$$\mathsf{E}\left[W_Q\right] = \frac{C_a^2 + C_s^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} \mathsf{E}[S]$$

Batching: $C_{sB}^2 = \dfrac{B\mathsf{V}[S_0] + \mathsf{V}[T]}{(B\,\mathsf{E}[S_0] + \mathsf{E}[T])^2}$

Nonpreemptive: $\mathsf{V}[S] = \mathsf{V}[S_0] + \dfrac{\mathsf{V}[T]}{B} + \dfrac{B-1}{B^2}(\mathsf{E}[T])^2$

Preemptive: $A = \dfrac{m_f}{m_r + m_f}, C_s^2 = C_0^2 + 2A(1-A)\dfrac{m_r}{\mathsf{E}[S_0]}$

$$C_{di}^2 = 1 + (1-\rho_i^2)(C_{ai}^2 - 1) + \frac{\rho_i^2}{\sqrt{c_i}}(C_{si}^2 - 1)$$

$$f_i(n_i) = \begin{cases} G(i)^{-1}(c_i\rho_i)^{n_i}(n_i!)^{-1}, & \text{if } n_i < c_i, \\ G(i)^{-1}c_i^{c_i}\rho_i^{n_i}(c_i!)^{-1}, & \text{if } n_i \geq c_i \end{cases}$$

$$\text{with } G(i) = \sum_{n=0}^{c_i-1} \frac{(c_i\rho_i)^n}{n!} + \frac{(c_i\rho_i)^{c_i}}{c_i!}\frac{1}{1-\rho_i}$$

$$\mathsf{E}[L_i] = \frac{(c_i\rho_i)^{c_i}}{c_i!G(i)}\frac{\rho_i}{(1-\rho_i)^2} + c_i\rho_i$$

$$f_i(n_i) = \frac{1}{\Pi_{k=1}^{n_i}\min\{k, c_i\}}\left(\frac{V_i}{\mu_i}\right)^{n_i}, i = 0 \ldots M$$

$$V_i = (VP)_i = \sum_{j=0}^{M} V_j P_{ji}$$

INDEX