Analysis of Queueing Systems with Sample Paths and Simulation

Nicky D. van Foreest

March 12, 2018

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 of more customers get his/her shopping for free. Of course, there are some constraints: at least one of the cashier facilities has to 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.) 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 making 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 some 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 on 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 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 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 house, 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 performance measures. Part of our work is to derive analytic models. 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. However, these models have severe shortcomings when it comes to analyzing real queueing systems, in particular when particular control rules have to be assessed. 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 cases simulation is a very useful and natural approach.

In the first part of these notes we concentrate on the analysis of *sample paths of a queueing process*. We 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, most 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 paths, and based on such sample paths we derive behavioral and statistical properties of the system. Thus, the performance measures defined for sample paths are precisely those used for simulation.

Our aim is not to provide rigorous proofs for all results derived below; for this we refer to El-Taha and Stidham Jr. [1998]. As a consequence we tacitly assume in the remainder that results derived from the (long-run) analysis of a particular sample path are equal to their 'probabilistic counterpart'.

In the second part of these notes 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. As such, theory, simulation and algorithms form a nicely round out part of work. For this part we refer to the book of Prof. Zijm; the present set of notes augments the discussion there.

Exercises I urge you to make *all* exercises in this set of notes. 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 many 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 problems should be doable, but hard.

The solution manual is meant to prevent you from getting stuck and to help you increase your knowledge of probability, linear algebra, programming (analysis with computer support), and queueing in particular. Thus, read the solutions very care-

fully.

As a guideline to making the exercises I recommend the following approach. First read the notes. Then attempt to make a exercises for 10 minutes or so by yourself. If by that time you have not obtained a good idea on how to approach the problem, check the solution manual. Once you have understood the solution, try to repeat the arguments *with the solution manual closed*.

Further reading The following books are very useful to extend one's knowledge of probability theory and queueing systems:

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

Acknowledgments I would like to acknowledge dr. J.W. Nieuwenhuis for our many discussions on the formal aspects of queueing theory and prof. dr. W.H.M. Zijm for allowing me to use the first few chapters of his book.

1 Single-Station Queueing Systems

In this chapter, we start with a discussion of the exponential distribution and the related Poisson process, as these concepts are perhaps the most important building blocks of queueing theory. With these concepts we can specify the arrival and service processes of customers, so that we can construct queueing processes and define performance measures to provide insight into the (transient and average) behavior of queueing processes. As it turns out, these queueing processes can often be easily implemented as computer programs, thereby allowing to use simulation to analyze queueing systems. We then continue with developing models for various single-station queueing systems in steady-state, which is, in a sense to be discussed later, the long-run behavior of a stochastic system. In the analysis we use sample-path arguments to count how often certain events occur as functions of time. Then we define probabilities in terms of limits of fractions of these counting processes. Another useful aspect of sample-path analysis is that the definitions for the performance measures are entirely constructive, hence by leaving out the limits, they provide expressions that can be right away used in statistical analysis of (simulations of) queueing systems. Level-crossing arguments will be of particular importance as we use these time and again to develop recursions by which we can compute steady-state probabilities of the queue length or waiting time process.

1.1 Poisson Distribution

Theory and Exercises

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 $N(s,t)$ for the number of customers that entered during a time interval of $(s,t]$. Clearly, as we do not know in advance how many customers will enter, we model $N(s,t)$ as a random variable for all times s and t.

Our first assumption is that the rate at which customers enter does not significantly change 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

$$\mathsf{E}[N(s,t)] = \lambda(t-s). \tag{1.1}$$

The constant λ is often called the *arrival rate*.

The second assumption is that $\{N(s,t), s \le t\}$ has *stationary* and *independent increments*. Stationarity means that the distribution of the number of arrivals are the same for all intervals of equal length. Formally, $N(s_1,t_1)$ has the same distribution as $N(s_2,t_2)$ if $t_2 - s_2 = t_1 - s_1$. Independence means, roughly speaking, that knowing that $N(s_1,t_1) = n$, does not help to make any predictions about $N(s_2,t_2)$ if the intervals (s_1,t_1) and (s_2,t_2) have no overlap.

To find the distribution of $N(t) = N(0,t)$, 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 second 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 such tiny 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 $\{B_i\}$ are independent. The total number of arrivals $N_n(t)$ that occur in n intervals is then binomially distributed

$$\mathsf{P}(N_n(t) = k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{1.2}$$

1.1.1. Show that $\mathsf{E}[N_n(t)] = \sum_{i=1}^{n} \mathsf{E}[B_i] = np$.

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

In fact, in some appropriate sense, $N_n(t)$ converges to $N(t)$ for $n \to \infty$ such that

$$\mathsf{P}(N(t) = k) = e^{-\lambda t} \frac{(\lambda t)^k}{k!}. \tag{1.3}$$

We say that $N(t)$ is *Poisson distributed* with rate λ, and write $N(t) \sim P(\lambda t)$.

1.1.3. Show how the binomial distribution (1.2) converges to the Poisson distribution (1.3) if $n \to \infty$, $p \to 0$ such that $np = \lambda$.

Thus, if we assume that the arrival rate of customers is constant for some amount of time, e.g., an hour in a shop, and we assume that the probability of a customer arriving in some very small amount of time, e.g., a millisecond, is small, the resulting number arrival of customers is Poisson distributed.

1.1.4. Show that if $N(t) \sim P(\lambda t)$, $\mathsf{E}[N(t)] = \lambda t$.

1.1.5. Show that if $N(t) \sim P(\lambda t)$, $\mathsf{V}[N(t)] = \lambda t$.

With moment generating functions we compute the mean and variance of $N(t)$ with less effort. Recall that the *moment generating function* of a random variable $X \geq 0$ and t a real number is defined as

$$M_X(t) = \mathsf{E}\left[e^{tX}\right].$$

1.1.6. Show that the moment generating function of a Poisson distributed random variable $N(t)$ is

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

1.1.7. Use the moment generating function of $N(t)$ to compute $\mathsf{E}[N(t)]$ and $\mathsf{V}[N(t)]$.

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

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

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

1.1.8. Show that the SCV of $N(t)$ is equal to $1/\lambda t$.

For the next couple of exercises, we need the concept of conditional probability. To help you recall this, consider the next question.

1.1.9. We have to give one present to one of three children. As we cannot divide the present 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 the number wins the present. Show that the probability of winning the present is the same for each child.

Below, and elsewhere, we use the following efficient notation: the function $f = o(h)$ means that f is such $f(h)/h \to 0$ as $h \to 0$. The next exercise helps you practice with this.

1.1.10. If c is a constant (in $\mathbb{R}$) and the functions $f, g = o(h)$, then $c \cdot f = o(h)$, $f + g = o(h)$, $f(h) \to 0$ as $h \to 0$, and $f \cdot g = o(h)$. (If you believe this right away, you may skip the solution. The proof is perhaps harder than you think.)

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

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

1.1.12. Show that if $N(t) \sim P(\lambda t)$, we have for small h, $\mathsf{P}(N(t+h) = n+1 \,|\, N(t) = n) = \lambda h + o(h)$.

1.1.13. Show that if $N(t) \sim P(\lambda t)$, we have for small h, $\mathsf{P}(N(t+h) \geq n+2 \,|\, N(t) = n) = o(h)$.

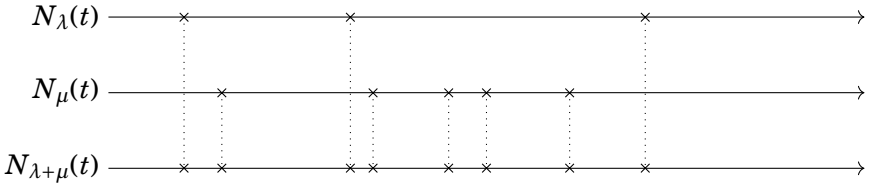
We define the Poisson *process* as the set $N = \{N(t), t \geq 0\}$. Observe that this process is a much more complicated object than the Poisson distributed random variable $N(t)$. The process is an uncountable set of random variables $N(t)$ for all $t \geq 0$, whereas $N(t)$ is just *one* random variable.

1.1.14. Show that for a Poisson process N,

$$\mathsf{P}(N(0,s] = 1 \,|\, N(0,t] = 1) = \frac{s}{t},$$

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

Merging, or the *superposition*, of Poisson processes occurs often in practice. We have two Poisson processes, for instance, the arrival processes N_λ of men and N_μ 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 to a woman, in lower line to an arrival of a general customer at the shop. Thus, the shop 'sees' the superposition $N_{\lambda+\mu}$ of these two arrival processes.



For the next set of exercises, assume that $N_\lambda(t) \sim \mathrm{P}(\lambda t)$, $N_\mu(t) \sim \mathrm{P}(\mu t)$ and are independent.

1.1.15. Show that $N_{\lambda+\mu}(t) = N_\lambda(t) + N_\mu(t) \sim \mathrm{P}((\lambda+\mu)t)$. Thus, when we merge two independent Poisson processes, the merged process is also a Poisson process whose rate is the sum of the rates of the original Poisson processes.

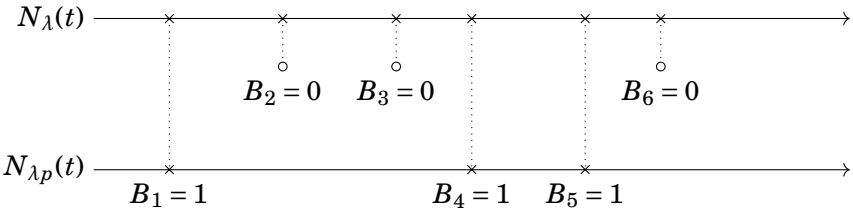
Again, there are two Poisson arrival processes. Let arrivals of the first type occur with rate λ, arrivals of the second type with rate μ. Given that an arrival occurred, the next question asks you to compute the probability that the arrival was of the first type.

1.1.16. Show that

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

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

Besides merging Poisson streams, we can also consider the concept of *splitting*, or *thinning*, a stream into sub-streams, as follows. When a customer arrives, throw a coin that lands heads with probability p and tails with $q = 1 - p$. When the coin lands heads, the customer is of type 1 (e.g., a woman), otherwise of type 2 (e.g., a child). Another way of thinning is by modeling the stream of people passing a shop as a Poisson process with rate λ. With probability p a person decides, independent of anything else, to enter the shop, c.f. the figure below. The crosses at the upper line are passersby in the street. The crosses at the lower line are the customers that enter the shop. The outcome of the kth throw of a coin is indicated by B_k. In the figure below, $B_1 = 1$ so that the first passerby turns into a customer entering the shop; the second passerby does not enter as $B_2 = 0$, and so on. Like this, the stream of passerby's is thinned with Bernoulli distributed random variables.



The concept of merging and thinning are also 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 to on another machine and the rest $(1 - p)$ of the jobs leaves the system. Then we can model the arrival stream at the second machine as a thinned stream (with probability p) of the departures of the first machine. Merging occurs where the output streams of various stations arrive at another station.

1.1.17. Show that the Poisson process obtained by thinning the original process is $\sim P(\lambda t p)$ for any t.

In fact, it can be proven that Poisson processes are the only arrival processes are such that, when merged or thinned, the resulting processes are again Poisson processes. Thus, if you believe that it is reasonable that a thinned processes has similar statistical properties as the original process, except that the mean becomes smaller, than the Poisson process is the only process that can be consistent with your intuition.

Hints

h.1.1.1 Use that $\mathsf{E}[X+Y] = \mathsf{E}[X] + \mathsf{E}[Y]$.

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

h.1.1.4 Apply the definition of the expectation, and in the computations use that

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

h.1.1.5 Realize that $\mathsf{V}[N] = \mathsf{E}\left[N^2\right] - (\mathsf{E}[N])^2$. Compute $\mathsf{E}\left[N^2\right]$, use Exercise 1.1.4 for $(\mathsf{E}[N])^2$ and the hint of Exercise 1.1.4 to simplify the calculations.

h.1.1.6 Observe first that for a random variable $X \in \mathbb{N}$ and some function f, we have that

$$\mathsf{E}[f(X)] = \sum_{n=0}^{\infty} f(n)\mathsf{P}(X=n).$$

Then take the function $f(x) = e^{sx}$. Finally, use that $\mathsf{P}(N(t)=k) = e^{-\lambda t}(\lambda t)^k/k!$

h.1.1.7 Recall that $M'_{N(t)}(0) = \mathsf{E}[N(t)]$ and $M''_{N(t)}(s) = \mathsf{E}\left[(N(t))^2\right]$.

h.1.1.9 For the second child, condition on the event that the first does not chose the right number. Use the definition of conditional probability:

$$\mathsf{P}(A|B) = \frac{\mathsf{P}(AB)}{\mathsf{P}(B)},$$

provided $\mathsf{P}(B) > 0$.

h.1.1.11 Recall the definition of the conditional probability $\mathsf{P}(A\,|B) = \mathsf{P}(AB)/\mathsf{P}(B)$, where we write $AB = A \cap B$. This definition only holds when $\mathsf{P}(B) > 0$.

Think about the meaning of the formula $\mathsf{P}(N(t+h) = n\,|\,N(t) = n)$. It is a conditional probability that should be read like this: given that up to time t we have seen n arrivals (i.e, $N(t) = n$), what is the probability that just a little later (at $t+h$) the number of arrivals is still n, i.e, $N(t+h) = n$? Then use the definition of the Poisson distribution to compute this probability. Finally, use Taylor's expansion of e^x to see that $e^x = 1 + x + o(x)$ for $|x| \ll 1$.

h.1.1.12 Use the previous exercise.

h.1.1.13 Use that $\sum_{i=2}^{\infty} x^i/i! = \sum_{i=0}^{\infty} x^i/i! - x - 1 = e^x - x - 1$.

h.1.1.14 Observe that

$$\{N(0,s]+N(s,t] = 1\} \cap \{N(0,s] = 1\} = \{1+N(s,t] = 1\} \cap \{N(0,s] = 1\} = \{N(s,t] = 0\} \cap \{N(0,s] =$$

Next use that these two events are independent.

h.1.1.15 Use a conditioning argument or use moment generating functions.

In particular, for conditioning, use that $\mathsf{P}(AB) = \mathsf{P}(A\,|\,B)\mathsf{P}(B)$. More generally, if the set A can be split into disjoint sets B_i, i.e, $A = \bigcup_{i=1}^{n} B_i$, then

$$\mathsf{P}(A) = \sum_{i=1}^{n} \mathsf{P}(AB_i) = \sum_{i=1}^{n} \mathsf{P}(A\,|\,B_i)\mathsf{P}(B_i),$$

where we use the conditioning formula to see that $\mathsf{P}(AB_i) = \mathsf{P}(A\,|\,B_i)\mathsf{P}(B_i)$. Now choose practical sets B_i.

In the computations, use the standard binomial formula $(a+b)^n = \sum_{i=0}^{b} \binom{n}{i} a^{n-i} b^i$.

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

h.1.1.17 There are (at least) two ways to get this result.

Method 1: Condition on the total number of arrivals $N(t) = n$ up to time t. Then, realize that the probability that a job is of type 1 is p. Hence when you consider n jobs in total, the number $N_1(t)$ of type 1 jobs is binomially distributed. Thus, given that n jobs arrived, the probability of k 'successes' (i.e., arrivals of type 1), is

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

Again use that if the set A can be split into disjoint sets B_i, i.e, $A = \bigcup_{i=1}^{n} B_i$, then

$$\mathsf{P}(A) = \sum_{i=1}^{n} \mathsf{P}(A\,|\,B_i)\mathsf{P}(B_i).$$

Now choose practical sets B_i.

Method 2: You might also consider the random variable

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

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

Solutions

s.1.1.1

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

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

s.1.1.3 Now we like to relate $N_n(t)$ and $N(t)$. It is clear that we at least want the expectations to be the same, that is, $np = \lambda t$. This implies that

$$p = \frac{\lambda t}{n}.$$

Substituting this in this in the expression for $\mathsf{P}(N_n(t) = k)$ gives

$$\mathsf{P}(N_n(t) = k) = \binom{n}{k}\left(\frac{\lambda t}{n}\right)^k \left(1 - \frac{\lambda t}{n}\right)^{n-k}.$$

To see that

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

use that

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

Observe now that, as λt is finite, $n/(n-\lambda t) \to 1$ as $n \to \infty$. Also for any finite k, $(n-k)/n \to 1$. Finally, we use that $(1 + x/n)^n \to e^x$ so that , $\left(1 - \frac{\lambda t}{n}\right)^n \to e^{-\lambda t}$. The rest is easy, so that, as $n \to \infty$, the above converges to

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

s.1.1.4 When a random variable N is Poisson distributed with parameter λ,

$$\begin{aligned}
\mathsf{E}[N] &= \sum_{n=0}^{\infty} n e^{-\lambda} \frac{\lambda^n}{n!} \\
&= \sum_{n=1}^{\infty} n e^{-\lambda} \frac{\lambda^n}{n!}, \text{ since the term with } n = 0 \text{ cannot contribute} \\
&= e^{-\lambda} \lambda \sum_{n=1}^{\infty} \frac{\lambda^{n-1}}{(n-1)!} \\
&= e^{-\lambda} \lambda \sum_{n=0}^{\infty} \frac{\lambda^n}{n!}, \text{ by a change of variable} \\
&= e^{-\lambda} \lambda e^{\lambda} \\
&= \lambda.
\end{aligned}$$

s.1.1.5

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

Hence, $\mathsf{V}[N] = \mathsf{E}\left[N^2\right] - (\mathsf{E}[N])^2 = \lambda^2 + \lambda - \lambda^2 = \lambda$.

s.1.1.6 Since $N(t)$ is Poisson distributed with parameter λt,

$$\begin{aligned}
M_{N(t)}(s) &= \mathsf{E}\left[e^{sN(t)}\right] \\
&= \sum_{k=0}^{\infty} e^{sk} \mathsf{P}(N(t) = k) \\
&\quad \text{where we use that } \mathsf{E}[f(N(t))] = \sum_{n=0}^{\infty} f(n) \mathsf{P}(N(t) = n) \\
&= \sum_{k=0}^{\infty} e^{sk} \frac{(\lambda t)^k}{k!} e^{-\lambda t}
\end{aligned}$$

$$= e^{-\lambda t} \sum_{k=0}^{\infty} \frac{(e^s \lambda t)^k}{k!}$$
$$= \exp(-\lambda t + e^s \lambda t) = \exp(\lambda t(e^s - 1)).$$

s.1.1.7 Using the expression for the moment generating function of the previous exercise,

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

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

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

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

$$\mathsf{V}[N(t)] = \mathsf{E}\left[(N(t))^2\right] - (\mathsf{E}[N(t)])^2 = \lambda t + (\lambda t)^2 - (\lambda t)^2 = \lambda t.$$

s.1.1.8

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

s.1.1.9 The probability that the first child to guess also wins is 1/3. What is the probability for child number two? Well, for him/her to win, it is necessary that child one does not win and that child two guesses the right number of the remaining numbers. Assume, without loss of generality that child 1 chooses 3 and that this is not the right number. Then

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

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

s.1.1.10 First,

$$\lim_{h \to 0} \frac{cf(h)}{h} = c \lim_{h \to 0} \frac{f(h)}{h} = 0, \text{ as } f = o(h)$$
$$\lim_{h} \frac{f(h) + g(h)}{h} = \lim_{h} \frac{f(h)}{h} + \lim_{h} \frac{g(h)}{h} = 0.$$

As a consequence, $f = o(h) \implies -f = o(h)$. Therefore $|f| = \max\{f, -f\} = o(h)$. Furthermore, letting $f^+ = \max\{f, 0\}$, we have that $2f^+ = |f| + f$, hence $f^+ = o(h)$. Likewise $f^- = f - f^+ = o(h)$. Then, if $h \in (0,1)$,

$$-f^-(h)/h \le f(h) \le f^+(h)/h,$$

we see that $f(h) \to 0$ since $f^-, f^+ = o(h)$. Finally, $f \cdot g = o(h)$, since

$$\lim_h \frac{f(h)g(h)}{h} = \lim_h f(h) \lim_h \frac{g(h)}{h} = 0.$$

s.1.1.11 Write $N(s,t]$ for the number of arrivals in the interval $(s,t]$. First we make a few simple observations: $N(t+h] = N(t) + N(t,t+h]$, hence

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

Thus,

$$\begin{aligned}\mathsf{P}(N(t+h) = n | N(t) = n) &= \mathsf{P}(N(t+h) = n, N(t) = n)/\mathsf{P}(N(t) = n) \\ &= \mathsf{P}(N(t,t+h] = 0, N(t) = n)/\mathsf{P}(N(t) = n) \\ &= \mathsf{P}(N(t,t+h] = 0)\,\mathsf{P}(N(t) = n)/\mathsf{P}(N(t) = n), \\ &\quad \text{(by independence of } N(t,t+h] \text{ and } N(t)) \\ &= \mathsf{P}(N(t,t+h] = 0) \\ &= \mathsf{P}(N(0,h] = 0) \\ &\quad \text{(by stationarity of the Poisson process)} \\ &= e^{-\lambda h}(\lambda h)^0/0! \\ &= e^{-\lambda h} = 1 - \lambda h + o(h).\end{aligned}$$

s.1.1.12

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

s.1.1.13

$$\mathsf{P}(N(t+h) \ge n+2 | N(t) = n) = \mathsf{P}(N(t,t+h] \ge 2)$$

$$
\begin{aligned}
&= e^{-\lambda h} \sum_{i=2}^{\infty} \frac{(\lambda h)^i}{i!} = e^{-\lambda h} \left(\sum_{i=0}^{\infty} \frac{(\lambda h)^i}{i!} - \lambda h - 1 \right) \\
&= e^{-\lambda h}(e^{\lambda h} - 1 - \lambda h) = 1 - e^{-\lambda h}(1 + \lambda h) \\
&= 1 - (1 - \lambda h + o(h))(1 + \lambda h) = 1 - (1 - \lambda^2 h^2 + o(h)) \\
&= \lambda^2 h^2 + o(h) = o(h).
\end{aligned}
$$

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

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

s.1.1.14 From the hint,

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

s.1.1.15 Choose the *sets* B_i as $\{N_\lambda(t) = i\}$. Then, the set $\{N_\lambda(t) + N_\mu(t) = n\}$ can be decomposed into the subsets B_i, i.e.,

$$
\{N_\lambda(t) + N_\mu(t) = n\} = \cup_{i=0}^{n} \{N_\lambda(t) = i\}.
$$

To see this, consider an example. Cats arrive at rate λ, dogs with rate μ. Suppose that 5 animals (cats plus dogs) arrived. Then there are 6 possibilities to form a set with 5 animals in total: 0 cats + 5 dogs, 1 cat + 4 dogs, ..., 5 cats + 0 dogs. The right-hand side of the equality decomposes the fact that 5 animals arrive into sets in which we know that the number of cats is $0, 1, 2, \ldots, 5$. Also, if we know the number of cats, we also know the number of dogs, as this is 5 minus the number of cats.

Now that we know how to conveniently split up the set $\{N_\lambda(t) + N_\mu(t) = n\}$, we can use conditioning:

$$
\mathsf{P}\left(N_\lambda(t) + N_\mu(t) = n\right) = \sum_{i=0}^{n} \mathsf{P}\left(N_\lambda(t) + N_\mu(t) = n | N_\lambda(t) = i\right) \mathsf{P}(N_\lambda(t) = i).
$$

Now, if $N_\lambda(t) = i$, then

$$N_\lambda(t) + N_\mu(t) = n \iff i + N_\mu(t) = n \iff N_\mu(t) = n - i.$$

With this,

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

Now with the moment generating functions of $N_\lambda(t)$ and $N_\mu(t)$, and using that by the independence of $N_\lambda(t)$ and $N_\mu(t)$, the moment generating function of $N_\lambda + N_\mu$ is the product of the moment generating functions of N_λ and N_μ,

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

Finally, since generating function uniquely characterizes the distribution, and since the above expression has the same form as a Poisson random variable with parameter $(\lambda + \mu)t$, we can conclude that $N(t) \sim P((\lambda + \mu)t)$.

s.1.1.16 With the above:

$$\begin{aligned}
&\mathsf{P}\left(N_\lambda(h) = 1 | N_\lambda(h) + N_\mu(h) = 1\right) \\
&= \frac{\mathsf{P}\left(N_\lambda(h) = 1, N_\lambda(h) + N_\mu(h) = 1\right)}{\mathsf{P}\left(N_\lambda(h) + N_\mu(h) = 1\right)} \\
&= \frac{\mathsf{P}\left(N_\lambda(h) = 1, N_\mu(h) = 0\right)}{\mathsf{P}\left(N_{\lambda+\mu}(h) = 1\right)} \\
&= \frac{\mathsf{P}\left(N_\lambda(h) = 1\right) \mathsf{P}\left(N_\mu(h) = 0\right)}{\mathsf{P}\left(N_{\lambda+\mu}(h) = 1\right)}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\lambda h \exp(-\lambda h)\exp(-\mu h)}{((\lambda+\mu)h)\exp(-(\lambda+\mu)h)} \\
&= \frac{\lambda h \exp(-(\lambda+\mu)h)}{((\lambda+\mu)h)\exp(-(\lambda+\mu)h)} \\
&= \frac{\lambda}{\lambda+\mu}.
\end{aligned}
$$

s.1.1.17 Method 1: Suppose that N_1 is the thinned stream, and N the total stream. Then, dropping the dependence on t for the moment,

$$
\begin{aligned}
\mathsf{P}(N_1 = k) &= \sum_{n=k}^{\infty} \mathsf{P}(N_1 = k, N = n) = \sum_{n=k}^{\infty} \mathsf{P}(N_1 = k \,|\, N = n)\mathsf{P}(N = n) \\
&= \sum_{n=k}^{\infty} \mathsf{P}(N_1 = k \,|\, N = n) e^{-\lambda}\frac{\lambda^n}{n!} \\
&= \sum_{n=k}^{\infty} \binom{n}{k} p^k (1-p)^{n-k} e^{-\lambda}\frac{\lambda^n}{n!}, \quad \text{by the hint} \\
&= e^{-\lambda}\sum_{n=k}^{\infty} \frac{p^k(1-p)^{n-k}}{k!(n-k)!}\lambda^n = e^{-\lambda}\frac{(\lambda p)^k}{k!}\sum_{n=k}^{\infty}\frac{(\lambda(1-p))^{n-k}}{(n-k)!} \\
&= e^{-\lambda}\frac{(\lambda p)^k}{k!}\sum_{n=0}^{\infty}\frac{(\lambda(1-p))^{n}}{n!} = e^{-\lambda}\frac{(\lambda p)^k}{k!}e^{\lambda(1-p)} \\
&= e^{-\lambda p}\frac{(\lambda p)^k}{k!}.
\end{aligned}
$$

We see that the thinned stream is Poisson with parameter λp. (For notational ease, we left out the t, otherwise it is $P(\lambda t p)$.

Method 2: Now consider $Y = \sum_{i=1}^{N} Z_i$. Suppose that $N = n$, so that n arrivals occurred. Then we throw n coins with success probability p. It is clear that Y is indeed a thinned Poisson random variable.

To obtain the distribution of Y, model the coin as a generic Bernoulli distributed random variable Z. We first need

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

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

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

Then, using conditioning on N,

$$
\mathsf{E}\left[e^{sY}\right] = \mathsf{E}\left[\,\mathsf{E}\left[e^{s\sum_{i=1}^{n} Z_i}\,\middle|\, N = n\right]\right] = \mathsf{E}\left[\,\mathsf{E}\left[\left(1 + p(e^s - 1)\right)^n \,\middle|\, N = n\right]\right]
$$

$$= \sum_{n=0}^{\infty} \left(1 + p(e^s - 1)\right)^n e^{-\lambda} \frac{\lambda^n}{n!} = e^{-\lambda} \sum_{n=0}^{\infty} \frac{(1 + p(e^s - 1))^n \lambda^n}{n!}$$
$$= e^{-\lambda} \exp(\lambda(1 + p(e^s - 1))) = \exp(\lambda p(e^s - 1)).$$

Thus, Y has the same moment generating function as a Poisson distributed random variable with parameter λp. Since moment-generating functions specify the distribution uniquely, $Y \sim P(\lambda p)$.

1.2 Exponential Distribution

Theory and Exercises

In the previous section we introduced the Poisson process as a natural model of the number of jobs arriving in during intervals of time. As we will see in the sections to come, the modeling and analysis of any queueing system is typically easier if we specificy the (probability) distribution of the interarrival times, i.e., the time between consecutive arrival epochs of jobs. A particular fruitful model for the distribution of the interarrival times is the exponential distribution. Here we show how this relates to the Poisson distribution, and we also derive many useful general probability concepts and apply these to the exponential distribution. Then we use simulation to provide yet further movivation for the use of the exponential distribution as a useful model for interarrival times of customers.

Let us assume that the interarrival times form a sequence $\{X_i\}$ of *independent and identically distributed (i.i.d.)* random variables, and let us write X for the generic random time between two successive arrivals. For many queueing systems, measurements of the interarrival times between consecutive arrivals show that it is reasonable to model an interarrival X as an *exponentially distributed* random variable, i.e.,

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

Moreover, the exponential distribution derives directly from the Poisson distribution. Observe that, if there are no arrivals in some interval $[0, t]$, then it must be that $N(t) = 0$. Hence, the first interarrival time X, i.e., the time until the first customer, must be larger than t. Therefore,

$$\mathsf{P}(X1 > t) = \mathsf{P}(N(t) = 0) = e^{-\lambda t} \frac{(\lambda t)^0}{0!} = e^{-\lambda t}.$$

Recall that the constant λ is called the arrival *rate*.

In the sequel we often write $X \sim \exp(\lambda)$ to mean that X is exponentially distributed with rate λ.

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

1.2.2. If the random variable $X \sim \exp(\lambda)$, show that its mean $\mathsf{E}[X] = \frac{1}{\lambda}$. Interpret this result.

1.2.3. If $X \sim \exp(\lambda)$, show that its second moment $\mathsf{E}\left[X^2\right] = \frac{2}{\lambda^2}$.

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

The above exercises can also be easily solved with the moment generating function of $X \geq 0$

$$M_X(t) = \mathsf{E}\left[e^{tX}\right] = \int_0^\infty e^{tx} \lambda e^{-\lambda x} \, \mathrm{d}x.$$

1.2.5. Why do we require that $t \in [0, \lambda)$ in the definition of $M_X(t)$?

1.2.6. What is $M_X(0)$?

1.2.7. If X is an exponentially distributed random variable with parameter λ, show that its moment generating function

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

1.2.8. Use the moment generating function to show that

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

1.2.9. Prove that the square coefficient of variation (SCV) of X is $C^2 = 1$.

Let us now show with simulation how the exponential distribution originates. Consider N people that regularly visit a shop. We assume that we can characterize the interarrival times $\{X_k^i, k = 1,2,\ldots\}$ of customer i by some distribution function, for instance the uniform distribution. Then, with $A_0^i = 0$ for all i, define

$$A_k^i = A_{k-1}^i + X_k^i = \sum_{j=1}^{k} X_j^i, \tag{1.4}$$

as the arrival moment of the kth visit of customer i. Now the shop owner 'sees' the superposition of the arrivals of all customers. One way to compute the arrival moments of all customers together is to put all the arrival times $\{A_k^i, k = 1,\ldots,n, i = 1,\ldots,N\}$ into one set, and sort these numbers in increasing order. This results in the (sorted) set of arrival times $\{A_k, k = 1,2,\ldots\}$ at the shop of all customers together. Taking $A_0 = 0$, then

$$X_k = A_k - A_{k-1}, \tag{1.5}$$

must be the interarrival time between the $k-1$th and kth customer at the shop. Thus, with this procedure, starting from interarrival times of individual customers, we can construct interarrival times as seen by the shop.

Suppose that we generate, by means of simulation, many interarrival times for a set of individual customers, compute the arrival times by (1.4), sort these, and compute with (1.5) the interarrival times $\{X_k\}$ of customers as seen by the shop. To plot the *empirical distribution function*, or the *histogram*, of $\{X_k\}$, we just count the number of interarrival times smaller than time t for any t. Then the emperical distribution of $\{X_k\}$ is defined as

$$\mathsf{P}_n(X \le t) = \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}[X_k \le t],$$

where the *indicator function* is $\mathbb{1}[X_k \le t] = 1$ if $X_k \le t$ and $\mathbb{1}[X_k \le t] = 0$ if $X_k > t$.

Let us now compare the probability density as obtained for several simulation scenarios to the density of the exponential distribution, i.e., to $\lambda e^{-\lambda t}$. As a first example, take $N = 1$ customer and let the computer generate $n = 100$ uniformly distributed numbers on the set $[4,6]$. Thus, the time between two visits of this customer is somewhere between 4 and 6 hours, and the average interarrival times

$\mathsf{E}[X] = 5$. In a second simulation we take $N = 3$ customers, and in the third, $N = 10$ customers.

The emperical distributions are shown, from left to right, in the three panels in Figure 1.1. The continuous curve is the graph of $\lambda e^{-\lambda x}$ where $\lambda = N/\mathsf{E}[X] = N/5$. (Recall from Exercise (1.2.2) that when 1 person visits the shop with an average interarrival time of 5 hours, the arrival rate is 1/5. Hence, when N customers visit the shop, each with an average interarrival time of 5 hours, the total arrival rate as seen by the shop must be $N/5$.)

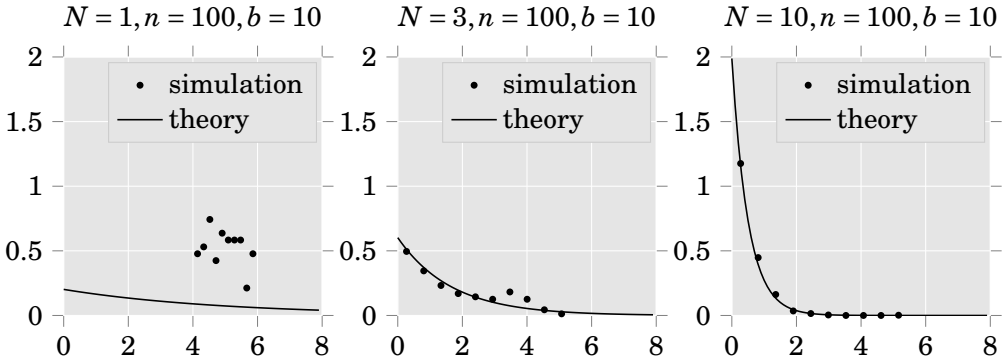


Figure 1.1: The interarrival process as seen by the shop owner. Observe that the dens $\lambda e^{-\lambda x}$ intersects the y-axis at level $N/5$, which is equal to the arrival rate when N perso visit the shop. The parameter $n = 100$ is the simulation length, i.e., the number of vis per customer, and $b = 10$ is number of bins to collect the data.

As a second example, we extend the simulation to $n = 1000$ visits to the shop, see Figure 1.2. In the third example we take the interarrival times to be normally distributed times with mean 5 and $\sigma = 1$, see Figure 1.3.

1.2.10. Try to make Figure 1.2 with simulation. (This is a very important exercise.)

As the graphs show, even when the customer population consists of 10 members, each visiting the shop with an interarrival time that is quite 'far' from exponential, the distribution of the interarrival times as observed by the shop is very well approximated by an exponential distribution. Thus, for a real shop, with many thousands of customers, or a hospital, call center, in fact for nearly every system that deals with random demand, it seems reasonable to use the exponential distribution to model

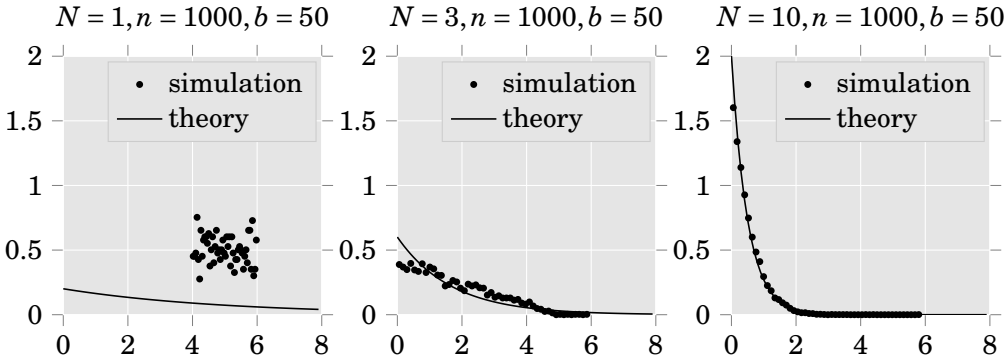


Figure 1.2: Each of the N customer visits the shop at uniformly distributed interarri
times, but now the number of visits is $n = 1000$.

interarrival times. In conclusion, the main conditions to use an exponential distribution are: 1) arrivals have to be drawn from a large population, and 2) each of the arriving customers decides, independent of the others, to visit the system.

We next provide further relations between the Poisson distribtuion and the exponential distribution.

1.2.11. Assume that the interarrival times $\{X_i\}$ are i.i.d. and $X_i \sim \exp(\lambda)$. Let $A_i = X_1 + X_2 + \cdots + X_i = \sum_{k=1}^{i} X_k$ with $i \geq 1$. Use the density of A_i or the moment generating function of A_i to show that

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

that is, the expected time to see i jobs is i/λ.

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

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

is the density of $A_i = X_1 + X_2 + \cdots + X_i = \sum_{k=1}^{i} X_k$ with $i \geq 1$.

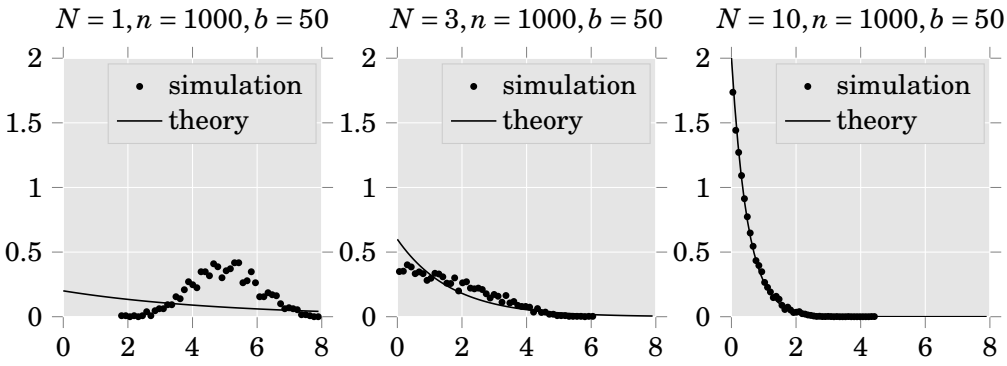


Figure 1.3: Each of the N customer visits the shop with normally distributed interarri
times with $\mu = 5$ and $\sigma = 1$.

1.2.13. Use the density f_{A_i} of the previous exercise to show that $\mathsf{E}[A_i] = i/\lambda$.

1.2.14. If the inter-arrival times $\{X_i\}$ are i.i.d. and exponentially distributed with mean $1/\lambda$, prove that the number $N(t)$ of arrivals during interval $[0,t]$ is Poisson distributed.

The relation between the Poisson process N and exponentially distributed interarrival times we discussed above is of paramount importance in the analysis of queueing processes: 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. and exponentially distributed with mean $1/\lambda$, in short,

$$X_i \sim \exp(\lambda) \Leftrightarrow N(t) \sim P(\lambda t).$$

Another very important concept is the following. 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. Thus, no matter how long we have been waiting for the next arrival to occur, the probability that it will occur in the next h seconds remains the same. This property seems to be vindicated also in practice: suppose that a patient with a broken arm just arrived at the emergency room of a hospital, what does that tell us about the time the next patient will be brought in? Not much, as most of us will agree.

1.2.15. Show that an exponentially distributed random variable is memoryless.

1.2.16. If $X \sim \exp(\lambda)$ and $S \sim \exp(\mu)$, and X and S are independent, show that

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

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

1.2.17. If $X \sim \exp(\lambda)$, $S \sim \exp(\mu)$, and X and S are independent, show that

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

In the sequel of this section we develop numerous important probability concepts, which we will also use many times in the rest of the course.

1.2.18. 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 $\mathsf{P}(S - X \leq u)$, for S and X independent.

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

$$\begin{aligned}\mathsf{E}[X] &= p\,\mathsf{E}[X_1] + q\,\mathsf{E}[X_2] \\ \mathsf{V}[X] &= p\,\mathsf{V}[X_1] + q\,\mathsf{V}[X_2] + pq(\mathsf{E}[X_1] - \mathsf{E}[X_2])^2.\end{aligned}$$

Interestingly, we see that even if $\mathsf{V}[X_1] = \mathsf{V}[X_2] = 0$, $\mathsf{V}[X] > 0$ if $\mathsf{E}[X_1] \neq \mathsf{E}[X_2]$. Bear this in mind; we will use these ideas later when we discuss the effects of

failures on the variance of service times of jobs.

Let B be a discrete random variable such that $\mathsf{P}(B = k) = f(k)$, where $f(k)$ is a given set of probabilities. We write

$$G(k) = \mathsf{P}(B > k) = \sum_{m=k+1}^{\infty} f(m),$$

for the *survivor function* of B. We can write this with an indicator function as

$$G(k) = \sum_{m=0}^{\infty} \mathbb{1}[m > k] f(m),$$

which makes the compution of certain expressions quite a bit easier.

1.2.20. Express the probability mass $f(k)$ and the survivor function $G(k)$ in terms of the distribution $F(k)$ of the batch size B. Which of the following is true: $G(k) = 1 - F(k)$, $G(k) = 1 - F(k-1)$, or $G(k) = 1 - F(k+1)$?

1.2.21. Use indicator functions to prove that $\sum_{k=0}^{\infty} G(k) = \mathsf{E}[B]$.

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

Let S be a continuous random variable S with distribution function F. We write

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

for the expection of X. Here $\mathrm{d}F(x)$ acts as a shorthand for $f(x)\mathrm{d}x$ when X has a density. [1]

1.2.23. Use indicator functions to prove that $\mathsf{E}[S] = \int_0^{\infty} x \, \mathrm{d}F = \int_0^{\infty} G(y)\mathrm{d}y$, where $G(x) = 1 - F(x)$.

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

1.2.24. Use indicator functions to prove that for a continuous random variable S with distribution function F, $\mathsf{E}\left[S^2\right] = \int_0^\infty x^2 \,\mathrm{d}F = 2\int_0^\infty yG(y)\,\mathrm{d}y$, where $G(x) = 1 - F(x)$.

1.2.25. Use integration by parts to show that for a continuous random variable S with distribution function F and survivor function $G = 1 - F$, $\int_0^\infty yG(y)\,\mathrm{d}y = \mathsf{E}\left[S^2\right]/2$.

1.2.26. Use that $\mathsf{E}[S] = \int_0^\infty x\,\mathrm{d}F = \int_0^\infty G(y)\,\mathrm{d}y$ to check that $\mathsf{E}[S] = \mu^{-1}$ if $F(x) = 1 - e^{-\mu x}$.

Hints

h.1.2.1 The interpretation of the function $1 - e^{-\lambda t}$ is a probability. What are the consequences of this? What happens if $\lambda = 0$?

h.1.2.2

$$\mathsf{E}[X] = \int_0^\infty tf(t)\,\mathrm{d}t = \int_0^\infty t\lambda e^{-\lambda t}\,\mathrm{d}t$$

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

h.1.2.3

$$\mathsf{E}\left[X^2\right] = \int_0^\infty t^2\lambda e^{-\lambda t}\,\mathrm{d}t = \frac{2}{\lambda^2}.$$

h.1.2.4 Use, and memorize, the very practical formula

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

h.1.2.7

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

h.1.2.8

$$\mathsf{E}[X] = M_X'(0) \qquad\qquad \mathsf{E}\left[X^2\right] = M_X''(0).$$

where $M'(t) = \frac{\mathrm{d}}{\mathrm{d}t}M(t)$.

h.1.2.11 What is $\mathsf{E}[X+Y]$ for some general random variables (each with finite mean)?

h.1.2.12 Check the result for $i=1$ by filling in $i=1$ (just to be sure that you have read the formula right), and compare the result to the exponential density. Then write $A_i = \sum_{k=1}^{i} X_k$, and compute the moment generating function for A_i and use that the interarrival times X_i are independent. Use the moment generating function of X_i.

h.1.2.13 Use the standard integral $\int_0^\infty x^n e^{-\alpha x}\,\mathrm{d}x = \alpha^{-n-1}n!$. Another way would be to use that, once you have the moment generating function of some random variable X, $\mathsf{E}[X] = \frac{\mathrm{d}}{\mathrm{d}t}M_X(t)|_{t=0}$.

h.1.2.14 Realize that $\mathsf{P}(N(t)=k) = \mathsf{P}(A_k \le t) - \mathsf{P}(A_{k+1} \le t)$.

h.1.2.15 Condition on the event $X > t$.

h.1.2.16 Use that if $Z = \min\{X,S\} > x$, it then must be that $X > x$ and $S > x$. Then use independence of X and S.

h.1.2.17 Define the joint distribution of X and S and carry out the computations, or use conditioning, or use the result of the previous exercise.

h.1.2.18 This is elementary, hence it might appear trivial, but it's not... In fact, I had a hard time finding a simple way to get the answer. It is good practice to try yourself before looking at the answer. Check also the previous problem, and make a drawing of the region over which you have to integrate.

h.1.2.19 Let X be the processing (or service) time at the server, and X_i the service time of a type i job. Then,

$$X = \mathbb{1}[T=1]X_1 + \mathbb{1}[T=2]X_2,$$

where $\mathbb{1}[]$ is the indicator function, that is, $\mathbb{1}[A]=1$ if the event A is true, and $\mathbb{1}[A]=0$ if A is not true.

h.1.2.20 This exercise is just meant to become familiar with the notation.

h.1.2.21 Write $\sum_{k=0}^{\infty} G(k) = \sum_{k=0}^{\infty}\sum_{m=k+1}^{\infty} \mathsf{P}(B=m)$, reverse the summations. Then realize that $\sum_{k=0}^{\infty} \mathbb{1}[k<m] = m$. You should be aware that this sort of problem is just a regular probability theory problem, nothing fancy. We use/adapt the tools you learned in calculus to carry out 2D integrals (or in this case 2D summations.)

h.1.2.22 $\sum_{i=0}^{\infty} iG(i) = \sum_{n=0}^{\infty} \mathsf{P}(B=n)\sum_{i=0}^{\infty} i\,1\{n \ge i+1\}$, and reverse the summations.

h.1.2.23 $\mathsf{E}[S] = \int_0^\infty x\,\mathrm{d}F = \int_0^\infty \int_0^\infty 1_{y\le x}\,\mathrm{d}y\,\mathrm{d}F(x)$.

h.1.2.24 $\int_0^\infty yG(y)\,\mathrm{d}y = \int_0^\infty y\int_0^\infty 1\{y\le x\}f(x)\,\mathrm{d}x\,\mathrm{d}y$.

Solutions

s.1.2.1 If $\lambda < 0$, then $1 - e^{-\lambda t}$ grows to $-\infty$ if $t \to \infty$. Just by itself this is not a problem. However, in our case $1 - e^{-\lambda t}$ has the interpretation of a distribution function. Now, recall that a distribution function is bounded to values in the interval $[0, 1]$.

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

s.1.2.2

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

For the interpretation, if jobs arrive at rate λ, the average time between two arrivals is λ^{-1}.

s.1.2.3

$$\begin{aligned}
\mathsf{E}\left[X^2\right] &= \int_0^\infty t^2\lambda e^{-\lambda t}\,\mathrm{d}t \\
&= \lambda^{-2}\int_0^\infty u^2 e^{-u}\,\mathrm{d}u, \quad \text{by change of variable } u = \lambda t, \\
&= -\lambda^{-2}\, t^2 e^{-t}\big|_0^\infty + 2\lambda^{-2}\int_0^\infty te^{-t}\,\mathrm{d}t \\
&= -2\lambda^{-2}\, te^{-t}\big|_0^\infty + 2\lambda^{-2}\int_0^\infty e^{-t}\,\mathrm{d}t \\
&= -2\lambda^{-2}\, e^{-t}\big|_0^\infty \\
&= 2/\lambda^2.
\end{aligned}$$

s.1.2.4 By the previous problems, $\mathsf{E}\left[X^2\right] = 2/\lambda^2$ and $\mathsf{E}[X] = 1/\lambda$.

s.1.2.5

$$M_X(t) = \mathsf{E}\left[e^{tX}\right] = \lambda\int_0^\infty e^{-(\lambda - t)x}\,\mathrm{d}x.$$

If $t - \lambda > 0$, then this integral becomes ∞.

s.1.2.6 $M_X(0) = \mathsf{E}\left[e^{0X}\right] = \mathsf{E}\left[e^0\right] = \mathsf{E}[1] = 1$

s.1.2.7

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

s.1.2.8 $\mathsf{E}[X] = M_X'(0) = \lambda/(\lambda - t)^2$. Hence, $M_X'(0) = 1/\lambda$. And, $\mathsf{E}\left[X^2\right] = M_X''(t) = 2\lambda/(\lambda - t)^3$, hence $\mathsf{E}\left[X^2\right] = M_X''(0) = 2\lambda/\lambda^3 = 2\lambda^{-2}$.

s.1.2.9 By the previous problems, $\mathsf{V}[X] = 1/\lambda^2$ and $\mathsf{E}[X] = 1/\lambda$.

s.1.2.10 The source code can be found in `progs/converge_to_exp.py`.

s.1.2.11 The simplest way to obtain the answer is to use that the expectation is a linear operator, i.e., $\mathsf{E}[X+Y] = \mathsf{E}[X] + \mathsf{E}[Y]$ for any r.v. X and Y. Then,

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

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

s.1.2.12 One way to find the distribution of A_i is by using the moment generating function $M_{A_i}(t) = \mathsf{E}\left[e^{tA_i}\right]$ of A_i. Let X_i be the interarrival time between customers i and $i-1$, and $M_X(t)$ the associated moment generating function. Using the i.i.d. property of the $\{X_i\}$,

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

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

s.1.2.13

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

Thus,

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

where we used the hint.

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

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

s.1.2.14 We want to show that

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

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

$$\begin{aligned}\mathsf{P}(A_{k+1}\le t) &= \lambda\int_0^t \frac{(\lambda s)^k}{k!}e^{-\lambda s}\,\mathrm{d}s\\ &= \lambda\frac{(\lambda s)^k}{k!}\frac{e^{-\lambda s}}{-\lambda}\Big|_0^t + \lambda\int_0^t\frac{(\lambda s)^{k-1}}{(k-1)!}e^{-\lambda s}\,\mathrm{d}s\\ &= -\frac{(\lambda t)^k}{k!}e^{-\lambda t} + \mathsf{P}(A_k\le t)\end{aligned}$$

We are done.

s.1.2.15 By the definition of conditional probability

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

As an aside, 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 want to check this, see e.g. Yushkevich and Dynkin [1969, Appendix 3].

s.1.2.16 Use that X and S are independent to get

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

s.1.2.17 There is more than one way to show that $\mathsf{P}(X \le S) = \lambda/(\lambda+\mu)$.

Method 1. Observe first that X and S, being exponentially distributed, have a density. Moreover, as they are independent, the joint density takes the form

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

With this,

$$\begin{aligned}\mathsf{P}(X \le S) &= \mathsf{E}[\mathbb{1}[X \le S]]\\ &= \int_0^\infty \int_0^\infty \mathbb{1}[x \le y] f_{X,S}(x,y)\,\mathrm{d}y\,\mathrm{d}x\\ &= \lambda\mu \int_0^\infty \int_0^\infty \mathbb{1}[x \le y] e^{-\lambda x}e^{-\mu y}\,\mathrm{d}y\,\mathrm{d}x\\ &= \lambda\mu \int_0^\infty e^{-\mu y} \int_0^y e^{-\lambda x}\,\mathrm{d}x\,\mathrm{d}y\\ &= \mu \int_0^\infty e^{-\mu y}(1-e^{-\lambda y})\,\mathrm{d}y\\ &= \mu \int_0^\infty (e^{-\mu y} - e^{-(\lambda+\mu)y})\,\mathrm{d}y\\ &= \mu \int_0^\infty (e^{-\mu y} - e^{-(\lambda+\mu)y})\,\mathrm{d}y\\ &= 1 - \frac{\mu}{\lambda+\mu}\end{aligned}$$

Method 2. You only have to study this if you (want to) know about conditioning with respect to a random variable with a density, otherwise, skip it. Start with the law of total probability in the discrete case:

$$\mathsf{E}[A] = \sum_{i=1}^{k} \mathsf{E}[A \mid B_i]\mathsf{P}(B_i),$$

where $B_1,\ldots,B_k$ forms a set of events such that $\cup_{i=1}^k B_i$ is equal to the sample space, $B_i \cap B_j = \varnothing$ and $\mathsf{P}(B_i) > 0$. Now in Capiński and Zastawniak [2003] it is shown that this can be extended to random variables X and Y with densities f_X, f_Y. The result is

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

Here, conceptually, $f_X(x)\mathrm{d}x \approx \mathsf{P}(X \in \mathrm{d}x)$ plays the role of $\mathsf{P}(B_i)$ in the discrete case above. With this notion, and observing that $f_S(s) = \mu e^{-\mu s}\,\mathrm{d}s$,

$$\mathsf{P}(X \le S) = \mathsf{E}[\,\mathbb{1}[X \le S]] = \int_0^\infty \mathsf{E}[\,\mathbb{1}[X \le S]|S = s]\,\mu e^{-\mu s}\,\mathrm{d}s.$$

Now, $\mathsf{E}[\,\mathbb{1}[X \le S]|S = s]$ is a tricky object, as $\mathsf{P}(S = s) = 0$, so that $\mathsf{E}[\,\mathbb{1}[X \le S]\,|\,S = s]$ *cannot* be defined as

$$\frac{\mathsf{E}[\,\mathbb{1}[X \le s]]}{\mathsf{P}(S = s)}.$$

The way to proceed is to consider the conditional probability density function of X given that $S = s$, which is defined as

$$f_{X|S}(x|s) = \frac{f_{X,S}(x,s)}{f_S(s)},$$

where, as before, $f_{X,S}(x,s)$ is the joint density of X and S. With this, we can properly define

$$\mathsf{P}(X \le S|S = s) = \mathsf{E}[\,\mathbb{1}[X \le S]|S = s] = \int_0^\infty \mathbb{1}[x \le s] f_{X|S}(x|s)\,\mathrm{d}x.$$

Using the definition of $f_{X|S}(x|s)$ and the independence of X and S it follows that

$$f_{X|S}(x|s) = \frac{f_{X,S}(x,s)}{f_S(s)} = \frac{\lambda e^{-\lambda x}\mu e^{-\mu s}}{\mu e^{-\mu s}} = \lambda e^{-\lambda x}$$

from which we get that

$$\begin{aligned}
\mathsf{E}[\,\mathbb{1}[X \le S]|S = s] &= \int_0^\infty \mathbb{1}[x \le s] f_{X|S}(x|s)\,\mathrm{d}x \\
&= \int_0^\infty \mathbb{1}[x \le s]\lambda e^{-\lambda x}\,\mathrm{d}x \\
&= \int_0^s \lambda e^{-\lambda x}\,\mathrm{d}x \\
&= 1 - e^{-\lambda s}.
\end{aligned}$$

And then,

$$\begin{aligned}
\mathsf{P}(X \le S) &= \int \mathsf{P}(X \le S|S = s)\,\mu e^{-\mu s}\,\mathrm{d}s \\
&= \mu \int_0^\infty \mathsf{E}[\,\mathbb{1}[X \le S]|S = s]\,e^{-\mu s}\,\mathrm{d}s = \mu \int_0^\infty (1 - e^{-\lambda s})e^{-\mu s}\,\mathrm{d}s.
\end{aligned}$$

Straightforward integration gives the final result.

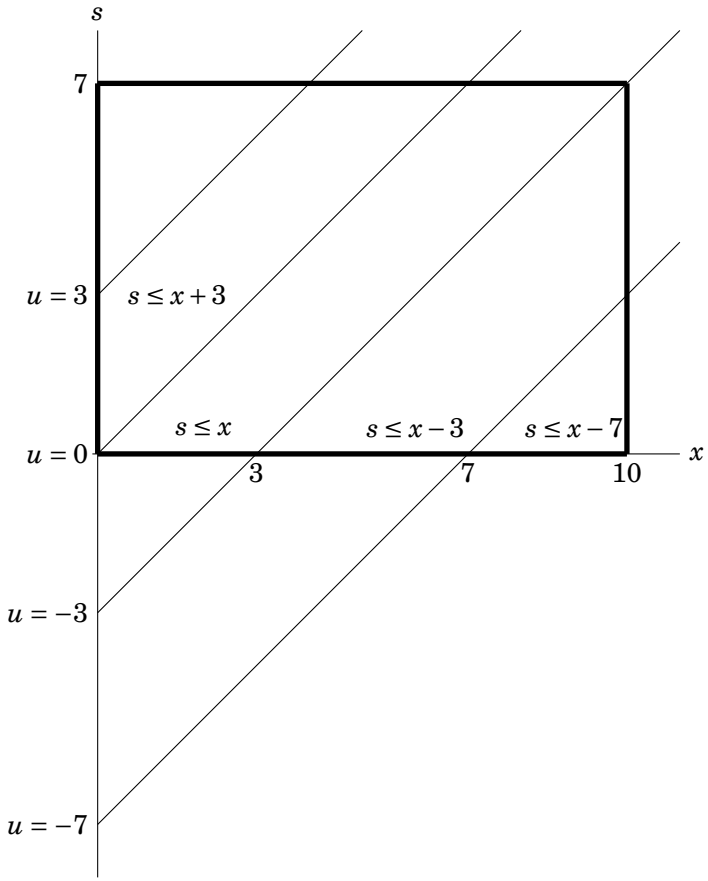
s.1.2.18 The joint density of S and X is given by

$$f_{XS}(x,y) = f_X(x) \cdot f_S(s) = \frac{1}{10}\,\mathbb{1}[0 \le x \le 10] \cdot \frac{1}{7}\,\mathbb{1}[0 \le s \le 7],$$

since X and S are independent. Thus,

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

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



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

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

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

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

where we multiply with 70 to get the normalization right.

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

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

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

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

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

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

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

```
\int_{0}^{10} \int_0^7 Boole[s<= x + u]  ds dx,
```

so, once you know LaTeX you can use Wolframalpha. Wolframalpha turned it to

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

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

s.1.2.19 With the hint,

$$\begin{aligned}
\mathsf{E}[X] &= \mathsf{E}[\mathbb{1}[T=1]X_1] + \mathsf{E}[\mathbb{1}[T=2]X_2] \\
&= \mathsf{E}[\mathbb{1}[T=1]]\,\mathsf{E}[X_1] + \mathsf{E}[\mathbb{1}[T=2]]\,\mathsf{E}[X_2], \text{ by the independence of } T, \\
&= \mathsf{P}(T=1)/\mu_1 + \mathsf{P}(T=2)/\mu_2 \\
&= p/\mu_1 + q/\mu_2 \\
&= p\,\mathsf{E}[X_1] + q\,\mathsf{E}[X_2].
\end{aligned}$$

(The next derivation may seem a bit long, but the algebra is standard. I include all steps so that you don't have to use pen and paper yourself if you want to check the result.) Next, using that

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

we get

$$\begin{aligned}
\mathsf{V}[X] &= \mathsf{E}\left[X^2\right] - (\mathsf{E}[X])^2 \\
&= \mathsf{E}\left[(\mathbb{1}[T=1]X_1 + \mathbb{1}[T=2]X_2)^2\right] - \left(\frac{p}{\mu_1} + \frac{q}{\mu_2}\right)^2 \\
&= \mathsf{E}\left[\mathbb{1}[T=1]X_1^2 + \mathbb{1}[T=2]X_2^2\right] - \left(\frac{p}{\mu_1} + \frac{q}{\mu_2}\right)^2 \\
&= p\,\mathsf{E}\left[X_1^2\right] + q\,\mathsf{E}\left[X_2^2\right] - \left(\frac{p}{\mu_1} + \frac{q}{\mu_2}\right)^2 \\
&= p\,\mathsf{V}[X_1] + p(\mathsf{E}[X_1])^2 + q\,\mathsf{V}[X_2] + q(\mathsf{E}[X_2])^2 - \left(\frac{p}{\mu_1} + \frac{q}{\mu_2}\right)^2 \\
&= p\,\mathsf{V}[X_1] + \frac{p}{\mu_1^2} + q\,\mathsf{V}[X_2] + \frac{q}{\mu_2^2} - \left(\frac{p}{\mu_1} + \frac{q}{\mu_2}\right)^2 \\
&= p\,\mathsf{V}[X_1] + q\,\mathsf{V}[X_2] + \frac{p}{\mu_1^2} + \frac{q}{\mu_2^2} - \frac{p^2}{\mu_1^2} - \frac{q^2}{\mu_2^2} - \frac{2pq}{\mu_1\mu_2} \\
&= p\,\mathsf{V}[X_1] + q\,\mathsf{V}[X_2] + \frac{p(1-p)}{\mu_1^2} + \frac{q(1-q)}{\mu_2^2} - \frac{2pq}{\mu_1\mu_2} \\
&= p\,\mathsf{V}[X_1] + q\,\mathsf{V}[X_2] + \frac{pq}{\mu_1^2} + \frac{qp}{\mu_2^2} - \frac{2pq}{\mu_1\mu_2} \\
&= p\,\mathsf{V}[X_1] + q\,\mathsf{V}[X_2] + pq(\mathsf{E}[X_1] - \mathsf{E}[X_2])^2.
\end{aligned}$$

s.1.2.20

$$f(k) = \mathsf{P}(B=k) = \mathsf{P}(B \le k) - \mathsf{P}(B \le k-1) = F(k) - F(k-1),$$

$$G(k) = \mathsf{P}(B > k) = 1 - \mathsf{P}(B \le k) = 1 - F(k).$$

It is all too easy to make, so called, off-by-one errors, such as in the three alternatives above. I nearly always check simple cases to prevent such simple mistakes. I advise you to acquire the same habit.

s.1.2.21 Observe first that $\sum_{k=0}^{\infty} \mathbb{1}[m > k] = m$, since $\mathbb{1}[m > k] = 1$ if $k < m$ and $\mathbb{1}[m > k] = 0$ if $k \ge m$. With this,

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

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

s.1.2.22

$$\begin{aligned}\sum_{i=0}^{\infty} iG(i) &= \sum_{i=0}^{\infty} i \sum_{n=i+1}^{\infty} \mathsf{P}(B = n) = \sum_{n=0}^{\infty} \mathsf{P}(B = n) \sum_{i=0}^{\infty} i 1\{n \ge i + 1\} \\ &= \sum_{n=0}^{\infty} \mathsf{P}(B = n) \sum_{i=0}^{n-1} i = \sum_{n=0}^{\infty} \mathsf{P}(B = n) \frac{(n-1)n}{2} \\ &= \sum_{n=0}^{\infty} \frac{n^2}{2} \mathsf{P}(B = n) - \frac{\mathsf{E}[B]}{2} = \frac{\mathsf{E}\left[B^2\right]}{2} - \frac{\mathsf{E}[B]}{2}.\end{aligned}$$

s.1.2.23

$$\begin{aligned}\mathsf{E}[S] &= \int_0^{\infty} x \, \mathrm{d}F = \int_0^{\infty} \int_0^{x} \mathrm{d}y \, \mathrm{d}F(x) \\ &= \int_0^{\infty} \int_0^{\infty} 1_{y \le x} \, \mathrm{d}y \, \mathrm{d}F(x) = \int_0^{\infty} \int_0^{\infty} 1_{y \le x} \, \mathrm{d}F(x) \, \mathrm{d}y \\ &= \int_0^{\infty} \int_y^{\infty} \mathrm{d}F(x) \, \mathrm{d}y = \int_0^{\infty} G(y) \, \mathrm{d}y\end{aligned}$$

s.1.2.24

$$\begin{aligned}\int_0^\infty yG(y)\,\mathrm{d}y &= \int_0^\infty y\int_y^\infty f(x)\,\mathrm{d}x\,\mathrm{d}y = \int_0^\infty y\int_0^\infty 1\{y\le x\}f(x)\,\mathrm{d}x\,\mathrm{d}y\\ &= \int_0^\infty f(x)\int_0^\infty y1\{y\le x\}\,\mathrm{d}y\,\mathrm{d}x = \int_0^\infty f(x)\int_0^x y\,\mathrm{d}y\,\mathrm{d}x\\ &= \int_0^\infty f(x)\frac{x^2}{2}\,\mathrm{d}x = \frac{\mathsf{E}\left[S^2\right]}{2}.\end{aligned}$$

s.1.2.25

$$\int_0^\infty yG(y)\,\mathrm{d}y = \frac{y^2}{2}G(y)\Big|_0^\infty - \int_0^\infty \frac{y^2}{2}g(y)\,\mathrm{d}y = \int_0^\infty \frac{y^2}{2}f(y)\,\mathrm{d}y = \frac{\mathsf{E}\left[S^2\right]}{2}, \tag{1.6}$$

since $g(y) = G'(y) = -F'(y) = -f(y)$.

s.1.2.26 If $F(x) = 1 - e^{-\mu x}$, we obtain that

$$\mathsf{E}[S] = \int_0^\infty e^{-\mu x}\,\mathrm{d}x = \mu^{-1}\int_0^\infty e^{-x}\,\mathrm{d}x = \mu^{-1}.$$

1.3 Kendall's Notation to Characterize Queueing Processes

Theory and Exercises

As will become apparent in Sections 1.4 and 1.5, the construction of any single-station queueing process involves three main elements: the distribution of the interarrival 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 interarrival 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 interarrival 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 interarrival times, B the distribution of the service times, c the number of servers, and K system size, i.e., the total number of customers that can be simultaneously present, whether in queue or in service. In this notation it is assumed that jobs are served in first-in-first-out (FIFO) order; FIFO scheduling is also often called first-come-first-serve (FCFS). To specify the distributions A and B often two letters are used: M to denote the exponential distribution as it is Memoryless, and G to denote any General distribution.

In the sequel we will often use Kendall's notation to distinguish between the different queueing models. Ensure that you familiarize yourself with this notation. Let us therefore illustrate this notation with some exercises.

1.3.1. What is the meaning of $M/M/1$?

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

1.3.3. What is the $D/D/1$ queue?

1.3.4. What is the meaning of $M/M/c$?

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

1.3.6. What is the meaning of $M/M/c/c$?

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

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

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

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

1.3.11. What is the meaning of $G/G/1$?

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

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

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

Hints

h.1.3.14 Look up the relevant definitions on wikipedia or Hall [1991]

Solutions

s.1.3.1 $M/M/1$: the distribution of the interarrival times is *M*emory-less, hence exponential, the service times are also *M*emoryless, and there is 1 server. As K is unspecified, the system can contain any number of jobs.

s.1.3.2 The interarrival times are exponentially distributed with rate λ; the service times are also exponential, but with parameter μ. Thus, if we know λ and μ, we have fully characterized the parameters of both distributions. Since the number of servers is 1, only λ and μ remain.

s.1.3.3 A queueing process with deterministic interarrival times and deterministic service times and 1 server.

s.1.3.4 $M/M/c$: A *multi-server* queue with c servers in which all servers have the same service capacity. Jobs arrive according to a Poisson process and have exponentially distributed processing times.

s.1.3.5

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

Note, sometimes, the K stands for the capacity in the queue, not the entire system. The notation differs among authors.

s.1.3.6 In this system the number of servers is the same as the system capacity, thus the queue length is always zero. This queueing system is useful to determine the number of beds in a hospital; the beds act as servers.

s.1.3.7 $M(n)/M(n)/1$: the interarrival times are exponential, just as the service times, but the rates of the arrival and service processes may depend on the queue length n.

s.1.3.8 $M^X/M/1$: Customers arrive with exponentially distributed interarrival times. However, each customer brings in a number of jobs, known as a batch. The number of jobs in each batch is distributed as the random variable X. Thus, the arrival process of work is *compound Poisson*.

s.1.3.9 $M/G/1$: the interarrival times are exponentially distributed, the service times can have any *G*eneral distribution (with finite mean), and there is 1 server.

s.1.3.10 $M/G/\infty$: exponential interarrival times, service times can have any distribution, and there is an unlimited supply of servers. This is also known as an *ample* server. Observe that in this queueing process, jobs actually never have to wait in queue; upon arrival there is always a free server available.

s.1.3.11 $G/G/1$: generally distributed interarrival and service times, 1 server.

s.1.3.12 $M/D/1-LIFO$. Now job service times are *D*eterministic, and the service sequence is last-in-first-out (LIFO).

s.1.3.13 Yes, take $G = D$ and $c = 1$.

s.1.3.14

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

1.4 Construction of Discrete-Time Queueing Processes

Theory and Exercises

To provide real-life motivation to analyze queueing systems we discuss a case. 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. Interestingly, the structure of the simulation is very simple so that it is also an exceedingly convincing tool to communicate the results of an analysis of a queueing system to managers (and the like).

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. Note that the length of such a period depends on the case 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.

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, which is useful since in many practical settings, e.g., production environments, it is easier to provide data in these terms than in terms of inter-arrival and service times.

Let us define

$$\begin{aligned} a_k &= \text{number of jobs that arrive at period } k, \\ c_k &= \text{the capacity, i.e., the maximal number of jobs that can be served, during period } k, \\ d_k &= \text{number of jobs that depart the queue } \textit{in} \text{ period } k, \\ Q_k &= \text{number of jobs in queue at the } \textit{end} \text{ of period } k. \end{aligned} \tag{1.7}$$

In the sequel we also call a_k the size of the batch arriving in period k. 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.

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 start and end of a period. Thus, the number of jobs in the system and in queue coincide; in this section 'queue length' henceforth means the number of jobs in the system.

Since Q_{k-1} is the queue length at the end of period $k-1$, it must also be the queue length 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 from the queue in period k is

$$d_k = \min\{Q_{k-1}, c_k\}, \tag{1.8a}$$

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

of departures, the queue at the end of period k is given by

$$Q_k = Q_{k-1} - d_k + a_k. \tag{1.8b}$$

1.4.1. Suppose that $c_k = 7$ for all k, and $a_1 = 5$, $a_2 = 4$ and $a_3 = 9$; also $Q_0 = 8$. What are d_k and Q_k for $k \geq 1$?

1.4.2. What are the consequences of setting $d_k = \min\{Q_{k-1} + a_k, c_k\}$ rather than the definition (1.8a)?

1.4.3. Implement Eqs. 1.8 in a computer program and simulate a simple single-server queueing system. (You should study the solution and the logic of the python code of this exercise closely; you don't have to memorize the code itself, though.)

Of course we are not going to carry out these computations by hand. Typically we use company data about the sequence of arrivals $\{a_k\}_{k=1,2,\ldots}$ and the capacity $\{c_k\}_{k=1,\ldots}$ and feed this data into a computer to carry out the recursions (1.8). 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.

We now show how to apply (1.7) and (1.8) to a real 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 is 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 that 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 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 after each other, 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 less in quiet weeks. Assuming that the system is stable, i.e., the average service capacity is larger 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.

To evaluate the effect of these suggestions on reducing the queueing dynamics we develop a simple simulator with which we can make a number of plots. As a first step we model the arrival process of patients as a Poisson process, c.f., Section 1.1. The duration of a period is taken to be a week. The average number of arrivals per period, based on data of the company, was slightly less than 12 per week; in the simulation we set it to $\lambda = 11.8$ per week. We model the capacity in the form of a matrix such that row i corresponds to the weekly capacity of psychiatrist i:

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

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

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

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

We next 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's 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 (1.8) 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 Figure 1.4. It is apparent that suggestions 1 and 2 above do not significantly affect the behavior of the queue length process.

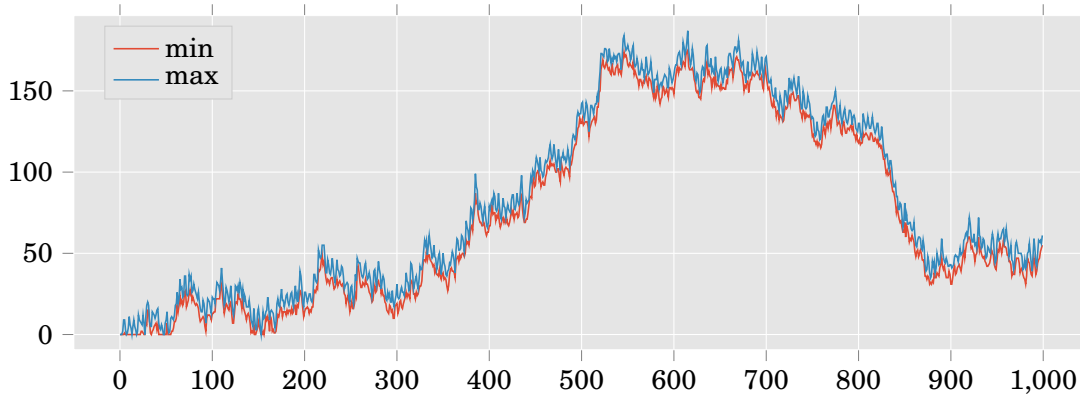


Figure 1.4: Effect of capacity and holiday plans. We plot for each time point the maximu and the minimum queue length for each of the policies. Apparently, the effect of each these policies is, for all practical purposes, negligible.

Now we consider suggestion 3, which comes down to doing more intakes when it is busy, and do less when it is quiet. A simple rule to implement this is by considering last week's queue Q_{n-1}: if $Q_{n-1} < 12$, i.e., the service capacity of one week, then

do e intakes less. Here, $e = 1$ or 2, or perhaps a larger number; it corresponds to the amount of control we want to exercise. When $Q_{n-1} > 24$, i.e., larger than two weeks of intakes, do e intakes more. 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.

Figure 1.5 shows a striking difference indeed. The queue does not explode any more, and already taking $e = 1$ has a large influence.

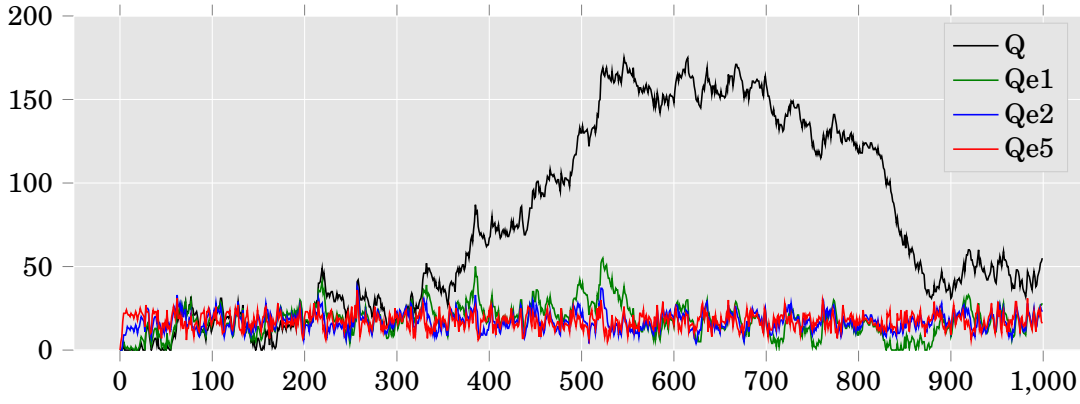


Figure 1.5: Controlling the number of intakes. Clearly, adapting the service rate 'd wonders' to control the queue length.

From this simulation experiment we learn that changing holiday plans or spreading the work over multiple servers, i.e., psychiatrists, does not significantly affect the queueing behavior. However, controlling the service rate as a function of the queue length improves the situation quite dramatically.

Observe that, even with these (deceitfully) simple recursions, we can obtain considerable insight into this, otherwise, very complicated controlled queueing process. (If the reader doubts the value of simulation, s/he should try to develop other mathematical methods to analyze multi-server queueing system with vacations, of which this case is an example.) As a matter of fact, with such simple recursions we can analyze many practical queueing situations. Together with students the author applied it numerous times, for instance,

- Should a certain hospital invest in a new MRI scanner to reduce waiting times?
- When to switch on and off a tin bath at an electronics component factory?

- How to route post parcels in a post sorting center.

In general, the study of queueing system 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 performance measures of relevance, 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 add service capacity. With simulation it is easy to study the effect of, hence evaluate, such decisions.

The reader should understand from the above case that, once we have the recursions, we can analyze the system and make plots to evaluate suggestions for improvement. Thus, getting the recursions is crucial to construct, i.e., model, queueing processes. For this reason, most of the exercises below focus on obtaining recursions for many different queueing systems.

Remark 1.4.1. A comment is required about the modeling exercises below. 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 the ones I make. I don't quite know how get out of this paradoxical situation. In a sense, to completely specify the model, we need the recursions. However, if the problem statement would contain the recursions, there would be nothing left to practice anymore. Another way is to make the problem description five times as long, but this is also undesirable. So, let's be pragmatic: the aim is that you practice with modeling, and that you learn from the solutions. If you obtain *reasonable* recursions, but they are different from mine, then your answer is just as good.

1.4.4. (Queue with Blocking) Consider 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?

1.4.5. (Estimating the lead time distribution.) Take $d_k = \min\{Q_{k-1} + a_k, c_k\}$, 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 $W_+(k)$

1.4.6. (Yield loss) A machine produces items, but a fraction p of the items produced in a period turns out to be faulty, and have to be made anew. Develop a set of recursions to cover this case.

1.4.7. (Rework) A machine produces items, but a fraction p of the items does not meet the quality requirements after the first service but need some extra service time but less than an entirely new arriving job. Make a model to analyze this case. Compare this case with the yield loss problem above.

Let's assume that the repair of a faulty requires half of the work of a new job, and that the faulty jobs are processed with priority over the new jobs. Also asume that faulty items do not need more than one repair (hence, faulty items that are repaired cannot be faulty anymore).

There are of course many different policies to treat rework. Another possibility is that faulty items are processed at the end of the day. Yet another possibility is that faulty items are collected until there are N, say, and then the entire batch of N is repaired.

1.4.8. (Cost models) A single-server queueing station processes customers. At the start of a period the server capacity is chosen, so that for period k the capacity is c_k. Demand that arrives in a period can be served in that period. It costs 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.

1.4.9. (N-policies) A machine can switch on and off. If the queue length hits N, the machine switches on, and if the system becomes empty, the machine switches off. It costs K to switch on the machine. There is also a cost 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.

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

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

1.4.12. (Priority queuing) Another 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.

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

1.4.14. (Queue with protected service capacity, lost capacity) Consider a single-server that serves two parallel queues. 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.

1.4.15. (Tandem networks) Consider a production network with two production stations in tandem, that is, the jobs processed by station A are in the next period to the downstream Station B. Extend the recursions of (1.8) to simulate this situation.

1.4.16. (A tandem queue with blocking) Consider a production network with two production stations in tandem with blocking: when intermediate queue, i.e., the queue front of Station B, exceeds some level M, then station A has to stop producing, and when $Q_k^B < M$ station A is not allowed to produce more than the intermediate queue can contain. Extend the recursions of (1.8) to simulate this situation.

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

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

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

1.4.20. (Inventory control) The recursions used in the exercises above can also be applied to analyze inventory control policies. Consider a production system that can produce maximally M_k items per week during normal working hours, and maximally N_k items during extra (weekend and evening) hours. Let, for

period k,

D_k = Demand in week k,
S_k = Sales, i.e., number of items sold, in week k,
r_k = Revenue per item sold in week k,
X_k = Number of items produced in week k during normal hours,
Y_k = Number of items produced in week k during extra hours,
c_k = Production cost per item during normal hours,
d_k = Production cost per item during extra hours,
h_k = holding cost per item, due at the end of week k,
I_k = On hand inventory level at the end of week k.

Management needs a production plan that specifies for the next T weeks the number of items to be produced per week. Formulate this problem as an LP problem, taking into account the inventory dynamics. Assume that demand must be met from on-hand inventory.

1.4.21. (Queue with setups) One server serves two parallel queues, one at a time. After serving one queue until it is empty, the server moves to the other queue. The change of queue requires one period setup time. (This is a tough problem; you can skip it if you don't have the time to really think about it.)

Hints

h.1.4.5 Consider a numerical example. Suppose $Q_{k-1} = 20$. Suppose that the capacity is $c_k = 3$ for all k. Then a job that arrives in the kth period, must wait at least 20/3 (plus rounding) periods before it can leave the system. Now generalize this numerical example.

h.1.4.20 Formulate the decision variables/controls, the objective and the constraints.

Solutions

s.1.4.1 $d_1 = 7$, $Q_1 = 8 - 7 + 5 = 6$, $d_2 = 6$, $Q_2 = 6 - 6 + 4 = 4$, $d_3 = 4$, $Q_3 = 4 - 4 + 9 = 9$, and so on.

s.1.4.2 The assumption is that the jobs arrive at the start of period k, before service in period k starts, rather than at the end of the period. Therefore the arrivals at period k can also be served during period k.

s.1.4.3 Here is an example in python; please read the code to see how I approach the problem. The code is really easy, as it is nearly identical to the mathematical specification. You don't have to memorize the specific syntax of the code, of course, but it is (at least I find it) interesting to see how little code is actually necessary to set things up.

Below I fix the seed of the random number generator to ensure that I always get the same results from the simulator. The arrival process is Poisson, and the number of services is fixed to 21 per period. I use `Q = np.zeros_like(a)` to make an array of the same size as the number of arrival a initially set to zeros. The rest of the code is nearly identical to the formulas in the text.

```
>>> import numpy as np

>>> np.random.seed(3) # fix the seed

>>> labda = 20
>>> mu = 21
```

These are the number of arrivals in each period.

```
>>> a = np.random.poisson(labda, 10)
>>> a
array([21, 17, 14, 10, 22, 22, 17, 17, 19, 21])
```

The number of potential services

```
>>> c = mu * np.ones_like(a)
>>> c
array([21, 21, 21, 21, 21, 21, 21, 21, 21, 21])
```

Now for the queueing recursions:

```
>>> Q = np.zeros_like(a)
>>> d = np.zeros_like(a)
>>> Q[0] = 10  # initial queue length
```

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

These are the departures and queue lengths for each period:

```
>>> d
array([ 0, 10, 17, 14, 10, 21, 21, 19, 17, 19])
>>> Q
array([10, 17, 14, 10, 22, 23, 19, 17, 19, 21])
```

Suppose we define loss as the number of periods in which the queue length exceeds 20. Of course, any other threshold can by taken. Counting the number of such periods is very easy in python: `(Q>20)` gives all entries of Q such $Q > 20$, the function `sum()` just adds them.

```
>>> loss = (Q > 20)
>>> loss
array([False, False, False, False,  True,  True, False, False, False,
        True])
>>> loss.sum()
3
```

Now all statistics:

```
>>> d.mean()
14.8
>>> Q.mean()
17.2
>>> Q.std()
4.377213725647858
>>> (Q > 20).sum()
3
```

Since this is a small example, the mean number of departures, i.e., `d.mean()`, is not equal to the arrival rate λ. Likewise for the computation of the mean, variance, and other statistical functions.

Now I am going to run the same code, but for a larger instance.

```
>>> num = 1000
>>> a = np.random.poisson(labda, num)
>>> c = mu * np.ones_like(a)
>>> Q = np.zeros_like(a)
>>> d = np.zeros_like(a)
>>> Q[0] = 10  # initial queue length

>>> for k in range(1, len(a)):
...     d[k] = min(Q[k - 1], c[k])
...     Q[k] = Q[k - 1] - d[k] + a[k]
...
>>> d.mean()
20.178
>>> Q.mean()
28.42
>>> Q.std()
10.155865300406461
>>> (Q > 30).sum()/num * 100
34.2
```

I multiply with 100 to get a percentage. Clearly, many jobs see a long queue, the mean is already some 24 jobs. For this arrival rate a service capacity of $\mu = 21$ is certainly too small.

Next, an example with the same arrival pattern but with a Poisson distributed number of jobs served per day, so the service rate is still 21, but the period capacity c is random variable with Poisson distribution with $P(21)$

```
>>> c = np.random.poisson(mu, num)
>>> Q = np.zeros_like(a)
>>> d = np.zeros_like(a)
>>> Q[0] = 10  # initial queue length

>>> for k in range(1, len(a)):
...     d[k] = min(Q[k - 1], c[k])
...     Q[k] = Q[k - 1] - d[k] + a[k]
...
>>> d.mean()
20.148
>>> Q.mean()
```

```
42.518
>>> Q.std()
22.841096208369684
>>> (Q > 30).sum()/num * 100
62.4
```

Compared to the case with constant service capacity, i.e., $c = 21$ in each period, we see that the average waiting time increases, just as the number of periods in which the queue length exceeds 30. Clearly, variability in service capacity does not improve the performance of the queueing system, quite on the contrary. In later sections we will see why this is so.

You can try to implement some of the other cases of the exercises in this section and analyze them in the same way. Hopefully you understand from this discussion that once you have the recursions to construct/simulate the queueing process, you are 'in business'. The rest is easy: make plots, do some counting, i.e., assemble statistics, vary parameters for sensitivity analysis, and so on.

s.1.4.4 All jobs that arrive such that the queue become larger than K must be dropped.

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

s.1.4.5 Let's tag the first customer that arrives in period k. This tagged customer sees Q_{k-1} customer in the system, hence the tagged customer's service can only start after all Q_{k-1} customers have been served. Now, if $Q_{k-1} - c_k > 0$, there are still people in front of the tagged customer. In fact, as long as $c_k + c_{k+1} + \cdots + c_m < Q_{k-1}$ the number of customers in front of the tagged customer are still in the system.

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

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

$$W_{-,k} := \max\left\{ m : \sum_{i=k}^{k+m} c_i < Q_{k-1} \right\},$$

and it must have been served before period

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

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

s.1.4.6 The amount produced in period k is d_k. Thus, pd_k is the amount lost, neglecting rounding errors for the moment. Thus, pd_k items have to be fed back to the system in the next period to be remade. Therefore the total amount of arrivals in period $k+1$ is $a'_{k+1} = a_{k+1} + pd_k$, i.e., the external arrivals plus the extra items. Now use the standard recursions but with the $\{a'_k\}$ rather than $\{a_k\}$.

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

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

s.1.4.7 Suppose again that a fraction p is faulty. Since these faulty items require less processing time than a new job, the service capacity c_k, i.e., the number of jobs that can be processed in period k, is a bit bigger; part of the capacity is spent on new jobs but another part is spent on the faulty jobs. By the assumptions above, the repair of a faulty requires half of the work of a new job, and the faulty jobs are processed with priority over the new jobs. Assume queue A contains the faulty items, and queue B the new jobs. Then the recursions become:

$$
\begin{aligned}
d_{k,A} &= \min\{Q_{k-1,A}, 2c_k\}, \text{ as faulty jobs require half of the processing time)} \\
c_{k,B} &= c_k - d_{k,A}/2, \\
d_{k,B} &= \min\{Q_{k-1,B}, c_{k,B}\}, \\
a_{k,A} &= pd_{k-1,B}, \\
Q_{k,A} &= Q_{k-1,A} + a_{k,A} - d_{k,A}, \\
Q_{k,B} &= Q_{k-1,B} + a_{k,B} - d_{k,B}.
\end{aligned}
$$

s.1.4.8 First consider the dynamics of the queue. Since the capacity is chosen at the start of the period:

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

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

$$
\frac{1}{T}\sum_{k=1}^{T}\left(\beta c_k + hQ_k\right).
$$

It is an interesting problem to find a policy that minimizes (the expectation of) this cost. The policy is such that the capacity for period k can be chosen based on

the queue length Q_{k-1} and *estimates* of the demands $\hat{d}_k, \hat{d}_{k+1}, \ldots$. This problem is not easy, as far as I can see.

s.1.4.9 First we need to implement the N-policy. For this we need an extra variable to keep track of the state of the server. Let $I_k = 1$ if the machine is on in period k and $I_k = 0$ if it is off. Then $\{I_k\}$ must satisfy the relation

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

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

$$I_{k+1} = \mathbb{1}[Q_k \geq N] + I_k \, \mathbb{1}[0 < Q_k < N] + 0 \cdot \mathbb{1}[Q_k = 0].$$

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

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

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

s.1.4.10 One model could be to let the server only switch on when the queue is larger than some threshold t, and when the server is on, it works at rate c per period. In that case, $c_k = c \, \mathbb{1}[Q_{k-1} > t]$.

s.1.4.11 Let c_k^i be the capacity allocated to queue i in period k. The fair rule gives that

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

Then,

$$d_k^1 = \min\{Q_{k-1}^1, c_k^1\},$$
$$Q_k^1 = Q_{k-1}^1 + a_k^1 - d_k^1,$$

and likewise for the other queue.

s.1.4.12 The rules below implement a strict priority rule for jobs type A, i.e., jobs sent into queue A.

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

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

s.1.4.13 First determine how much capacity queue B minimally needs in period k:

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

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

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

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

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

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

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

s.1.4.14 Queue A can use all capacity, except what is reserved for queue B:

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

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

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

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

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

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

s.1.4.15 Let a_k be the external arrivals at station A. Then:

$$\begin{aligned} d_k^A &= \min\{Q_{k-1}^A, c_k^A\}, \\ Q_k^A &= Q_{k-1}^A - d_k^A + a_k. \end{aligned} \tag{1.9}$$

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

$$\begin{aligned} a_k^B &= d_k^A, \\ d_k^B &= \min\{Q_{k-1}^B, c_k^B\}, \\ Q_k^B &= Q_{k-1}^B - d_k^B + a_k^B. \end{aligned} \tag{1.10}$$

s.1.4.16

$$\begin{aligned} d_k^A &= \min\{Q_{k-1}^A, c_k^A, M - Q_{k-1}^B\}, \\ Q_k^A &= Q_{k-1}^A - d_k^A + a_k, \\ a_k^B &= d_k^A, \\ d_k^B &= \min\{Q_{k-1}^B, c_k^B\}, \\ Q_k^B &= Q_{k-1}^B - d_k^B + a_k^B. \end{aligned} \tag{1.11}$$

This is a bit subtle: since there is room $M - Q_{k-1}^B$ at the intermediate buffer and $d_k^A \leq M - Q_{k-1}^B$, we know that in the worst case, i.e., when $c_k^B = 0$, still $Q_k^B = Q_{k_1}^B + d_k^A$.

Thus, we are sure that the queue length of the intermediate queue will not exceed M.

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

$$d^A_k = \min\{Q^A_{k-1}, c^A_k, \max\{M - Q^B_{k-1}, 0\}\}.$$

s.1.4.17 Realize that Stations A and B have their own arrivals.

$$\begin{aligned}
d^A_k &= \min\{Q^A_{k-1}, c^A_k\},\\
Q^A_k &= Q^A_{k-1} - d^A_k + a^A_k,\\
d^B_k &= \min\{Q^B_{k-1}, c^B_k\},\\
Q^B_k &= Q^B_{k-1} - d^B_k + a^B_k,\\
a^C_k &= d^A_k + d^B_k,\\
d^C_k &= \min\{Q^C_{k-1}, c^C_k\},\\
Q^C_k &= Q^C_{k-1} - d^C_k + a^C_k.
\end{aligned} \tag{1.12}$$

s.1.4.18 The behavior of the queue length process is easy:

$$\begin{aligned}
d_k &= \min\{Q_{k-1} + a^a_k + a^b_k, c\},\\
Q_k &= Q_{k-1} + a^a_k + a^b_k - d_k.
\end{aligned}$$

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

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

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

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

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

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

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

Similarly, for the b jobs must be

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

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

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

I currently lack time to check whether the above does not contain any off-by-one errors. For future work, implement it in a simulator...

s.1.4.19

1. Realize that the recursions of Eq (1.8) applied to the queueing situation at the first machine provide us with the total number of departures d_k during period k. However, it does not tell us about the type of these departures. Thus, to compute the queue in front of station A, we need do know the number of departures of type A, rather than the total number of departures of the first station.

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

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

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

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

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

s.1.4.20 The decision variables are X_k, Y_K and S_k (note, it is not necessary to meet all demand: the production cost and profit may vary per period.) The objective is

$$\max \sum_{k=1}^{T} (r_k S_k - c_k X_k - d_k Y_k - h_k I_k).$$

The constraints are

$$\begin{aligned} 0 \le S_k \le D_k, \\ 0 \le X_k \le M_k, \\ 0 \le Y_k \le N_k, \\ I_k = I_{k-1} + X_k + Y_k - S_k. \\ I_k >= 0. \end{aligned}$$

s.1.4.21 We need an extra variable p_k that specifies which queue is being served. If $p_k = 0$, the server is moving from one queue to the other, if $p = 1$, the server is at queue 1, and if $p_k = 2$ it is at queue 2. Then, with

$$\begin{aligned} d_k^1 = \min\{Q_{k-1}^1, c_k^1 \mathbb{1}[p_k = 1]\}, \\ d_k^2 = \min\{Q_{k-1}^2, c_k^2 \mathbb{1}[p_k = 2]\}, \end{aligned}$$

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

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

1.5 Construction of the G/G/1 Queueing Process in Continuous Time

Theory and Exercises

In the previous section we considered time in discrete 'chunks', minutes, hours, days, and so on. For given numbers of arrivals and capacity per period we use a set of recursions (1.8) to compute the departures and queue length per period. Another way to construct a queueing system is to consider inter-arrival times between consecutive customers and the service times each of these customers require. With this we obtain a description of the queueing system in continuous time. The goal of this section is to develop a set of recursions for the $G/G/1$ single-server queue.

Assume we are given, as basic data, the *arrival process* $\{A(t); t \geq 0\}$: 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 obtain various other interesting concepts, such as the arrival times of individual jobs. Specially, 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

$$A_k = \inf\{t : A(t) \geq k\}, \tag{1.13}$$

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

$$X_k = A_k - A_{k-1}. \tag{1.14}$$

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

$$A(t) = \sum_{k=1}^{\infty} \mathbb{1}[A_k \leq t], \tag{1.15a}$$

where $\mathbb{1}[]$ is the indicator function. Thus, in the above we count all arrivals that occur up to time t. Another, equivalent, way to define $A(t)$ is

$$A(t) = \sup\{k : A_k \leq t\}. \tag{1.15b}$$

Clearly, we see that from the inter-arrival times $\{X_k\}$ it is possible to construct $\{A_k\}$ and $\{A(t)\}$, and the other way around, from $\{A(t)\}$ we can find $\{A_k\}$ and $\{X_k\}$.

Figure 1.6 shows these relations graphically. To memorize it may be helpfull to write it like this:

$$A_k : \mathbb{N} \to \mathbb{R}, \quad \text{job id (integer) to arrival time (real number)},$$
$$A(t) : \mathbb{R} \to \mathbb{N}, \quad \text{time (real number) to number of jobs (integer)}.$$

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

1.5.2. Would it make sense to define $A(t)$ as $\inf\{k : A_k \geq t\}$?

To compute the departure times $\{D_k\}$ we proceed in stages. The first stage is to construct the *waiting time in queue* $\{W_{Q,k}\}$ as seen by the arrivals. In Figure 1.6 observe that the waiting time of the kth arrival must be equal to the waiting time of the $k-1$th customer plus the amount of *service time* required by job $k-1$ minus the time that elapses between the arrival of job $k-1$ and job k, unless the server becomes idle between jobs $k-1$ and k. In other words,

$$W_{Q,k} = [W_{Q,k-1} + S_{k-1} - X_k]^+, \tag{1.16}$$

where $[x]^+ = \max\{x, 0\}$. If we set $W_{Q,0} = 0$, we can compute $W_{Q,1}$ from this formula, and then $W_{Q,2}$ and so on.

The time job k *leaves the queue and moves on to the server* is

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

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 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

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

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

$$W_k = D_k - A_k = \tilde{A}_k + S_k - A_k = W_{Q,k} + S_k, \tag{1.17}$$

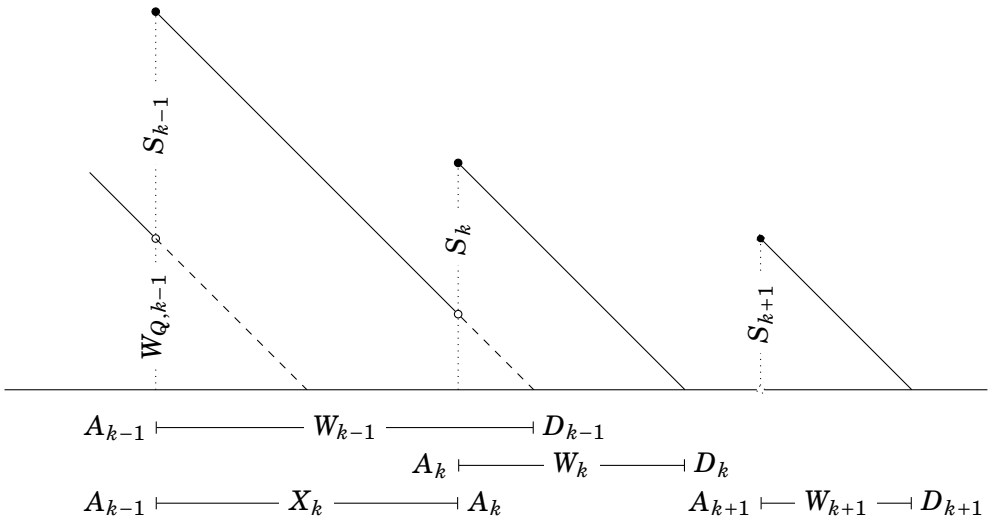


Figure 1.6: Construction of the $G/G/1$ queue in continuous time. The sojourn time of the kth job is sum of the work in queue $W_{Q,k}$ at its arrival epoch A_k and its service time S_k; its departure time is then $D_k = A_k + W_k$. The waiting time of job k is clearly equal to $W_{k-1} - X_k$. We also 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

where each of these equations has its own interpretation.

A bit of similar reasoning gives another recursion for W_k:

$$W_{Q,k} = [W_{k-1} - X_k]^+, \qquad W_k = W_{Q,k} + S_k = [W_{k-1} - X_k]^+ + S_k, \tag{1.18}$$

from which follows a recursion for D_k

$$D_k = A_k + W_k. \tag{1.19}$$

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

$$D(t) = \max\{k; D_k \le t\} = \sum_{k=1}^{\infty} \mathbb{1}[D_k \le t].$$

1.5.3. Assume that $X_1 = 10$, $X_2 = 5$, $X_3 = 6$ and $S_1 = 17$, $S_2 = 20$ and $S_3 = 5$,

compute the arrival times, waiting times in queue, the sojourn times and the departure times for these three customers.

1.5.4. 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 $\mathsf{P}(S_k = 1) = \mathsf{P}(S_k = 2)$. If $W_{Q,0} = 3$, what are the distributions of $W_{Q,1}$ and $W_{Q,2}$?

Another set of recursions to compute the arrival and departure times for the single server queue is the following:

$$\begin{aligned} A_k &= A_{k-1} + X_k, \\ D_k &= \max\{A_k, D_{k-1}\} + S_k, \\ W_k &= D_k - A_k. \end{aligned} \tag{1.20}$$

1.5.5. Why do the recursions Eq. (1.20) work?

1.5.6. Implement the above recursions in excel or some other computer program such as R, python, or julia, and check the results of the previous exercise.

Once we have the arrival and departure processes it is easy to compute the *number of jobs in the system* at time t as

$$L(t) = A(t) - D(t) + L(0), \tag{1.21}$$

where $L(0)$ is the number of jobs in the system at time $t = 0$; typically we assume that $L(0) = 0$. Thus, if we were to plot $A(t)$ and $D(t)$ as functions of t, then the difference $L(t)$ between the graphs of $A(t)$ and $D(t)$ tracks the number in the system, see Figure 1.7.

Observe that in a queueing system, jobs can be in queue or in service. For this reason we distinguish between the number in the system $L(t)$, the number in queue $L_Q(t)$, and the number of jobs in service $L_s(t)$. If we know $\tilde{A}(t)$, i.e. the number of jobs that departed from the queue up to time t, then

$$L_Q(t) = A(t) - \tilde{A}(t)$$

must be the number of jobs in queue. The above expressions for $L(t)$ and $L_Q(t)$ then show that the number in service must be

$$L_s(t) = \tilde{A}(t) - D(t) = L(t) - L_Q(t).$$

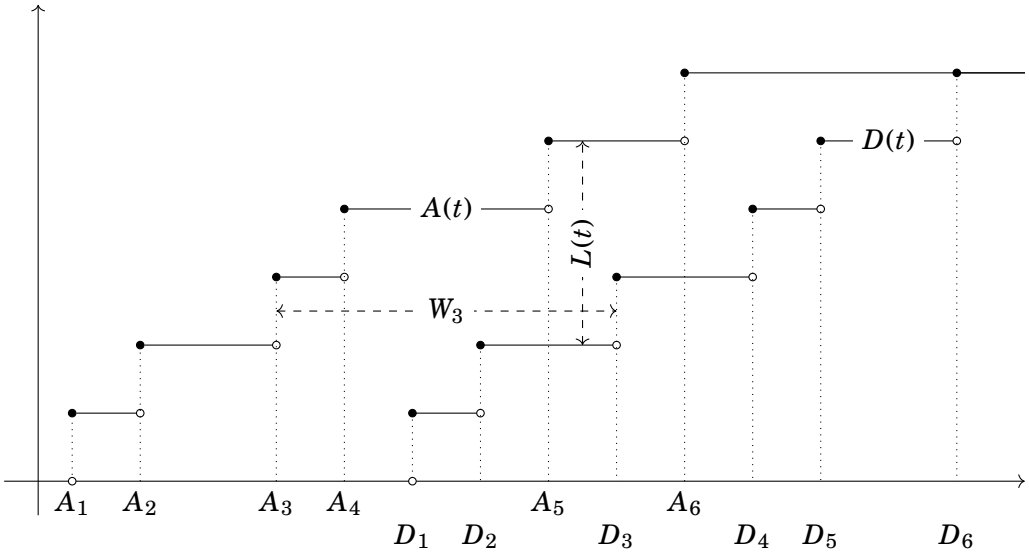


Figure 1.7: Relation between the arrival process $\{A(t)\}$, the departure process $\{D(t)\}$, t number in the system $\{L(t)\}$ and the waiting times $\{W_k\}$.

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

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

1.5.9. Why is $L(t) = L_Q(t) + L_s(t)$?

1.5.10. Express $L_Q(t)$ and $L_s(t)$ in terms of $A(t)$, $\tilde{A}(t)$ and $D(t)$.

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

1.5.12. Show that $L(t) = A(t) - D(t)$ and the system starts empty, implies that $L(t) = \sum_{k=1}^{\infty} \mathbb{1}[A_k \le t < D_k]$.

1.5.13. Show that $L(A_k) > 0 \implies A_k \le D_{k-1}$.

1.5.14. Define the number in the system as seen by the kth arrival as

$$L(A_k-), \tag{1.22}$$

where $L(A_k-) = \lim_{h \downarrow 0} L(A_k - h)$. (Observe that the kth job arrives at time A_k and $\{L(s)\}$ is right-continuous.)

Can you derive the following (algorithmic efficient) procedure to compute the number of jobs in the system as seen by arrivals,

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

Why do we take $i = k-1-L(A_{k-1}-)$ in the sum, and not $i = k-2-L(A_{k-1}-)$?

Finally, 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. Thus, the lines with slope -1 in Figure 1.6 show (a sample path of) the virtual waiting time.

Observe that, just as in Section 1.4, we have obtained a set of recursions by which we can run a simulation of a queueing process of whatever length we need, provided we have a sequence of have inter-arrival times $\{X_k\}$ and service times $\{S_k\}$. A bit of experimentation with computer programs such as R or python will reveal that this is easy.

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

1.5.16. Try to extend the recursions (1.20) to a situation in which one queue is served by two servers. (This is a hard problem, only for the interested.)

1.5.17. Suppose one server serves two queues, such that jobs in queue A are served with priority over jobs in queue B. Assume that service is not-preemptive. Can you develop a similar set of recursions for each of the queues, or, otherwise, an algorithm? Can you extend this to cover the LIFO (Last-In-First-Out) queue? (This is a hard problem; only for the interested.)

Hints

h.1.5.2 Compare this to the definition in (1.15).

h.1.5.3 The intent of this exercise is to make you familiar with the notation.

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

h.1.5.4 Use Eq. (1.18).

h.1.5.12 Use Boolean algebra. Write, for notational ease, $A = \mathbb{1}[A_k \le t]$ and $\bar{A} = 1 - A = \mathbb{1}[A_k > t]$, and define something similar for D. Then show that $A - D = A\bar{D} - \bar{A}D$, and show that $\bar{A}D = 0$. Finally sum over k.

h.1.5.13 Use that $L(A_k) > 0$ means that the system contains at least one job at the time of the kth arrival, and that $A_k \le D_{k-1}$ means that job k arrives before job $k-1$ departs.

h.1.5.15 Make a plot of the function $A_{A(t)} - t$. What is the meaning of $V(A_{A(t)})$? What is $V(A_{A(t)} + A_{A(t)} - t$?

h.1.5.16 The problem is that in a multi-server queueing systems, unlike for single-server queues, jobs can overtake each other: a small job that arrives after a very large job can still leave the system earlier. After trying for several hours, I obtained an inelegant method. A subsequent search on the web helped a lot. The solution below is a modification of N. Krivulin, 'Recursive equations based models of queueing systems'.

Solutions

s.1.5.1 $A(t)$ is the number of arrivals during $[0,t]$. Suppose that $A(t)=n$. This nth job arrived at time A_n. Thus, $A_{A(t)}$ is the arrival time of the last job that arrived before or at time t. In a similar vein, A_n is the arrival time of the nth job. Thus, the number of arrivals up to time n, i.e., $A(A_n)$, must be n.

s.1.5.2 Suppose $A_3=10$ and $A_4=20$. Take $t=15$. Then $\min\{k: A_k \geq 15\}=4$ since $A_3<t=15<A_4$. On the other hand $\max\{k: A_k \leq t\}=3$. And, indeed, at time $t=15$, 3 jobs arrived, not 4. So defining $A(t)$ as $\min\{k: A_k \geq t\}$ is not ok.

s.1.5.3 Lets feed it to the computer. Mind that in python (just like in C, and so on), arrays start at index 0, not at index 1.

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

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

>>> D = [0, 0, 0, 0]
>>> for i in range(1, len(X)):
...     D[i] = A[i] + WQ[i] + S[i]
...
>>> D
[0, 27, 47, 52]
```

s.1.5.4 First find the distribution of $Y_k := S_{k-1}-X_k$ so that we can write $W_{Q,k}=[W_{Q,k-1}+Y_k]^+$. Use independence of $\{S_k\}$ and $\{X_k\}$:

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

Dropping the dependence on k for ease, we get

$$\begin{aligned}
\mathrm{P}(Y=-2) &= \mathrm{P}(S-X=-2) = \mathrm{P}(S=1,X=3) = \mathrm{P}(S=1)\,\mathrm{P}(X=3) = \frac{1}{4}\\
\mathrm{P}(Y=-1) &= \mathrm{P}(S=2)\,\mathrm{P}(X=3) = \frac{1}{4}\\
\mathrm{P}(Y=0) &= \mathrm{P}(S=1)\,\mathrm{P}(X=1) = \frac{1}{4}\\
\mathrm{P}(Y=1) &= \mathrm{P}(S=2)\,\mathrm{P}(X=1) = \frac{1}{4}.
\end{aligned}$$

With this

$$\begin{aligned}
\mathrm{P}\left(W_{Q,1}=1\right) &= \mathrm{P}\left(W_{Q,0}+Y_0=1\right) = \mathrm{P}(3+Y_0=1) = \mathrm{P}(Y_0=-2) = \frac{1}{4}\\
\mathrm{P}\left(W_{Q,1}=2\right) &= \mathrm{P}(3+Y=2) = \mathrm{P}(Y=-1) = \frac{1}{4}\\
\mathrm{P}\left(W_{Q,1}=3\right) &= \mathrm{P}(3+Y=3) = \mathrm{P}(Y=0) = \frac{1}{4}\\
\mathrm{P}\left(W_{Q,1}=4\right) &= \mathrm{P}(3+Y=4) = \mathrm{P}(Y=1) = \frac{1}{4}.
\end{aligned}$$

And, then

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

Typing the solution becomes boring..., let's use the computer.

```
>>> from lea import Lea

>>> W = 3
>>> S = Lea.fromVals(1,  2)
>>> S
1 : 1/2
2 : 1/2
```

```
>>> X = Lea.fromVals(1,  3)
>>> X
1 : 1/2
3 : 1/2

>>> # This is WQ1
>>> W = Lea.fastMax(W + S - X, 0)
>>> W
1 : 1/4
2 : 1/4
3 : 1/4
4 : 1/4

>>> # This is WQ2
>>> W = Lea.fastMax(W + S - X, 0)
>>> W
0 : 3/16
1 : 3/16
2 : 4/16
3 : 3/16
4 : 2/16
5 : 1/16
```

Great! Our handiwork matches with the computer's results.

s.1.5.5 Of course, the service of job k cannot start before it arrives. Hence, it cannot leave before $A_k + S_k$. Therefore it must be that $D_k \geq A_k + S_k$. But the service of job k can also not start before the previous job, i.e. job $k-1$, left the server. Thus job k cannot start before D_{k-1}. To clarify it somewhat further, define S'_k as the earliest start of job k. Then it must be that $S'_k = \max\{A_k, D_{k-1}\}$—don't confuse the earliest start S'_k and the service time S_k—and $D_k = S'_k + S_k$.

s.1.5.6 You can check the files on `github` in the `progs` directory.

- `mm1.py` is the python implementation
- `mm1.R` is the R implementation
- `mm1.jl` is the julia implementation

Take your pick, and start playing with it. These examples are meant to be simple to understand, not necessarily super efficient.

s.1.5.7 Because $D_s(t) = D(t)$. Once customers leave the server, their service is completed, and they leave the queueing system.

s.1.5.8 All customers that left the system must have left the queue. Thus, $\tilde{A}(t) \geq D(t)$.

s.1.5.9 $L_Q(t) = A(t) - \tilde{A}(t)$. $L_S(t) = \tilde{A}(t) - D(t)$. This is in line with the fact that $L(t) = L_Q(t) + L_s(t) = A(t) - D(t)$.

s.1.5.10 $L_Q(t) = A(t) - \tilde{A}(t)$. $L_S(t) = \tilde{A}(t) - D(t)$. This is in line with the fact that $L(t) = L_Q(t) + L_s(t) = A(t) - D(t)$.

s.1.5.11 In this case, there are servers idling while there are still customers in queue. If such events occur, we say that the server is not work-conservative.

s.1.5.12

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

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

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

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

s.1.5.13 In a sense, the claim is evident, for, if the system contains a job when job k arrives, it cannot be empty. But if it is not empty, then at least the last job that arrived before job k, i.e., job $k-1$, must still be in the system. That is, $D_{k-1} \geq A_k$.

A more formal proof proceeds along the following lines. Using that $A(A_k) = k$ and $D(D_{k-1}) = k-1$,

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

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

s.1.5.14 Let $L(A_k-)$, i.e., the number of jobs in the system as 'seen by' job k. It must be that $L(A_k-) = k - 1 - D(A_k)$. To see this, assume first that no job has departed when job k arrives. Then job k must see $k-1$ jobs in the system. In general, if at time A_k the number of departures is $D(A_k)$, then the above relation for $L(A_k-)$ must hold. Applying this to job $k-1$ we get that $L(A_{k-1}-) = k - 2 - D(A_{k-1})$.

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

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

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

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

s.1.5.15 There is a funny way to do this. Recall from a previous exercise that if $A(t) = n$, then A_n is the arrival time of the nth job. Thus, the function $A_{A(t)}$ provides us with arrival times as a function of t. When $t = A_{A(t)}$, i.e., when t is the arrival time of the $A(t)$th job, we set $V(t) = V(A_{A(t)}) = W_{A(t)}$ i.e., the virtual waiting time at the arrival time $t = A_{A(t)}$ is equal to the waiting time of the $A(t)$th job. Between arrival moments, the virtual waiting time decreases with slope 1, until it hits 0. Thus,

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

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

s.1.5.16 The recursions for the two-server system are this:

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

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

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

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

but it is not correct. Can you find a counter example?

s.1.5.17 This question came up in class. I don't think this can be constructed as a straightforward recursion, and a search on the web lead to nothing. In case you can find a recursion, please let me know.

While a straightforward recursion may not exist, it is not really difficult to code this queueing discipline. The key idea is to put all jobs into one queue, but sort the elements in the queue in order of priority. Every time the server becomes empty, it checks the head of the queue. If the queue is empty, it wait until the next arrival occurs. Otherwise, it starts the service of the first job in line. When a new job

arrives, then identify its priority first, and then put it at the end of the jobs of the same priority.

This type of sorting is known as lexicographic sorting, which is what we do when we build a dictionary. First we sort words in order of first letter, then the second letter, and so on. In case of sorting the queue, we first have to sort in order of priority, then, within the class of jobs with the same priority, sort in ascending order of arrival time.

The python code is like this, where I include the FIFO case for comparison. As always, it is not obligatory to memorize the code.

```
class Fifo(Queue):
    def __init__(self):
        self.queue = SortedSet(key = lambda job: job.arrivalTime)

class Priority(Queue): # a priority queue
    def __init__(self, numServers = 1):
        self.queue = SortedSet(key = lambda job: job.p)
```

Interestingly, once we have a proper environment to carry out simulations, changing the queueing discipline is a one-liner!.

With the above, an algorithm for the LIFO is straightforward. Rather then sorting the jobs in queue in ascending order of arrival time, just sort them in descending order of arrival time.

```
class Lifo(Queue):
    def __init__(self):
        self.queue = SortedSet(key = lambda job: -job.arrivalTime)
```

Another interesting queueing rule is to sort in increasing job size;

```
class SPTF(Queue): # shortest processing time first
    def __init__(self):
        self.queue = SortedSet(key = lambda job: job.serviceTime)
```

Do you see how sort in descending order of job size (it just a matter of putting a minus sign at the right place)?

1.6 Queueing Processes as Regulated Random Walks

Theory and Exercises

In the construction of queueing processes as set out in Section 1.4 we are given two sequences of i.i.d. random variables: the number of arrivals $\{a_k\}$ per period and the service capacities $\{c_k\}$. Assuming that jobs can be served in the period they arrive, the departure and queue length processes are generated by the recursions

$$\begin{aligned} Q_k &= [Q_{k-1} + a_k - c_k]^+, \\ d_k &= Q_{k-1} + a_k - Q_k, \end{aligned} \tag{1.23}$$

where $[x]^+ := \max\{x, 0\}$.

1.6.1. What is the difference between the recursive schemes of (1.23) and (1.8)? Explain in the former scheme the formula for d_k.

Observe now that the relation for Q_k shares a resemblance to a random walk $\{Z_k, k = 0, 1, \ldots\}$ with Z_k given by

$$Z_k = Z_{k-1} + a_k - c_k. \tag{1.24}$$

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 $\{Q_k\}$ is restricted to the non-negative integers. In this section we show how to build the queueing process $\{Q_k\}$ from the random walk $\{Z_k\}$ by 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.

1.6.2. Show that Q_k satisfies the relation

$$Q_k = Z_k - \min_{0 \le i \le k} Z_i \wedge 0, \tag{1.25}$$

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

This recursion for Q_k leads to really interesting graphs. In Figure 1.8 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 Figure 1.9, $a_k \sim B(0.49)$ and the random walk is constructed as

$$Z_k = Z_{k-1} + 2a_k - 1. \tag{1.26}$$

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

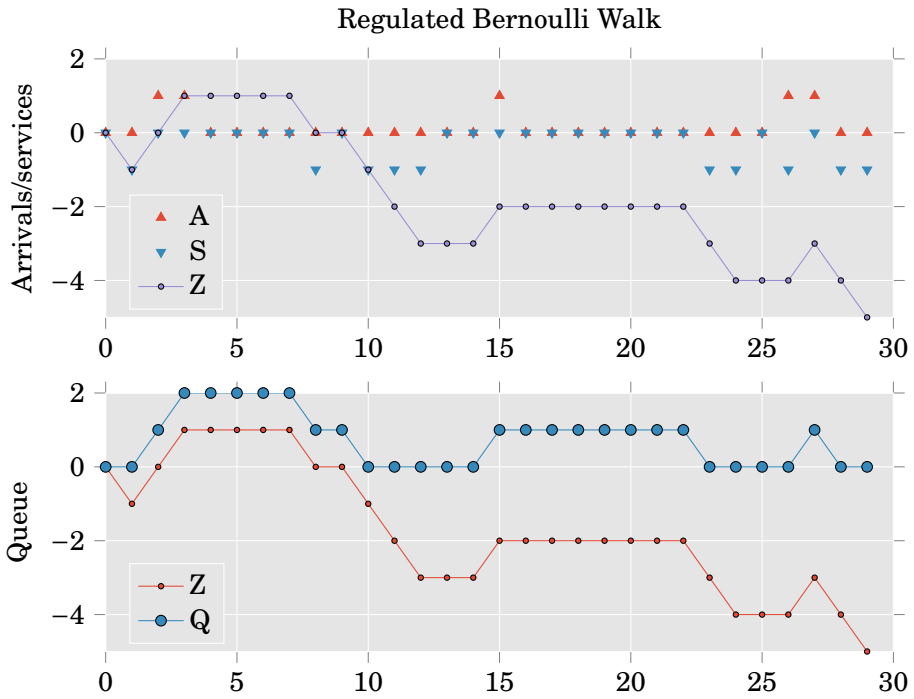


Figure 1.8: The upper panel shows a graph of the random walk Z. An upward pointi triangle corresponds to an arrival, a downward triangle to a potential service. The lov panel shows the queueing process $\{Q_k\}$ as a random walk with reflection.

With (1.25), we see that a random walk $\{Z_k\}$ can be converted into a queueing process $\{Q_k\}$, and we might try to understand the transient behavior of the latter by investigating the transient behavior of the former. For this, we first relate the random walk of the type (1.26) to a random walk in continuous time.

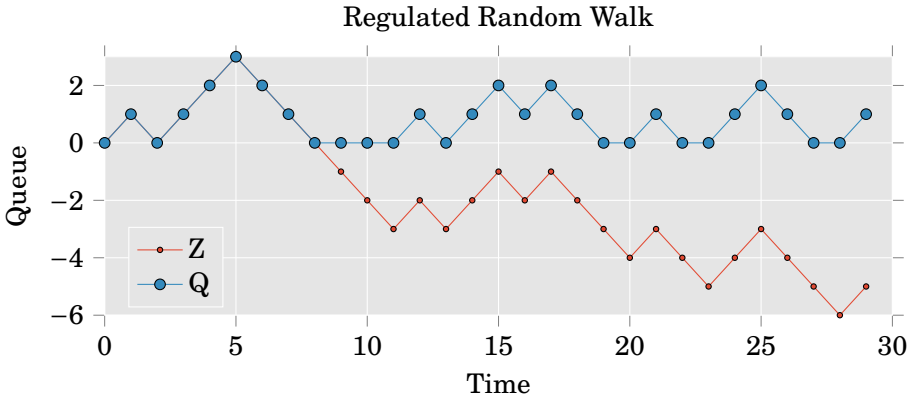


Figure 1.9: Another example of a reflected random walk.

1.6.3. Let $N_{\lambda+\mu}$ be a Poisson process with rate $\lambda+\mu$. If $\{a_k\}$ is an i.i.d. sequence of Bernoulli random variables such that $\mathsf{P}(a_k=1)=\lambda/(\lambda+\mu)=1-\mathsf{P}(a_k=0)$, show that the random variable

$$N_\lambda(t)=\sum_{k=1}^{\infty} a_k\,\mathbb{1}[k\le N_{\lambda+\mu}(t)],$$

has a Poisson distribution with rate λt.

Similarly, let

$$N_\mu(t)=N_{\lambda+\mu}(t)-N_\lambda(t)=\sum_{k=1}^{\infty}(1-a_k)\,\mathbb{1}[N_{\lambda+\mu}(t)\le k];$$

but this is $N_{\lambda+\mu}(t)$ thinned by the Bernoulli random variables $\{1-a_k\}$. Let $N_\lambda=\{N_\lambda(t)\}$ and $N_\mu=\{N_\mu(t)\}$ be the associated Poisson processes.

With the processes N_λ and N_μ constructed above from the sequence $\{a_k\}$ and the Poisson process $N_{\lambda+\mu}$ we can define the process $Z=\{Z(t)\}$ such that

$$Z(t)=Z(0)+N_\lambda(t)-N_\mu(t).$$

Thus, we let N_λ correspond to job arrivals and N_μ to departures. Observe that the times $\{T_k\}$ at which Z makes jumps are such that T_k-T_{k-1} have exponential distribution with mean $1/(\lambda+\mu)$. At the jump times, $Z(T_k)=Z_k$, where Z_k satisfies (1.26) with $\mathsf{P}(a_k=1)=\lambda/(\lambda+\mu)$. We call Z the *free* $M/M/1$ queue as, contrary to the real $M/M/1$ queue, Z can take negative values.

1.6.4. Show that

$$\mathsf{P}_m(Z(t)=n) = e^{-(\lambda+\mu)t}\left(\frac{\lambda}{\mu}\right)^{(n-m)/2} \sum_{k=0}^{\infty} \frac{(t\sqrt{\lambda\mu})^{2k+m-n}}{k!(k+m-n)!},$$

where $\mathsf{P}_m(\cdot)$ means that the random walk starts at m, i.e., $Z(0)=m$.

As an aside, the summation includes negative factorials when $k+m-n<0$. The tacit assumption is to take $n!\in\{\pm\infty\}$ for $n\in\mathbb{Z}_-$. Another way to get around this problem is to take $k=\max\{0,m-n\}$..

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 Q from Z via (1.25)), our hopes to finding anything simple for the transient analysis of the $M/M/1$ queue should not be too high. And the $M/M/1$ is but the simplest queueing system; other queueing systems will be more complicated yet. We therefore give up the analysis of such transient queueing systems and we henceforth contend ourselves with the analysis of queueing systems in the limit as $t\to\infty$. This of course warrants two questions: what type of limit is actually meant here, and what is the rate of convergence to this limiting situation? We address these questions subsequently.

The *long-run limiting behavior* of a queueing system (i.e., the first question) is an important topic by itself. The underlying question is what happens if we simulate the system for a long time. For instance, does there exist a random variable Q such that $Q_k\to Q$ in some sense? The answer to this question is in the affirmative, provided some simple stability conditions are satisfied, see Section 1.7. However, it requires a considerable amount of mathematics to make this procedure precise. To sketch what has to be done, first, we need to define $\{Q_k\}$ as random variables in their own right. Note that up to now we just considered each Q_k as a *number*, i.e., a measurement or simulation of the queue length time of the kth period. Defining Q_k as a random variable is not as simple as the definition of, for instance, the number of arrivals $\{a_k\}$; these random variables can be safely *assumed* to be i.i.d. However, the queue lengths $\{Q_k\}$ are certainly not i.i.d., but, as should be apparent from Eq. (1.18), they are *constructed* in terms of recursions. Next, based on these recursions, we need to show that the sequence of distribution functions $\{G_k\}$ associated with the random variables $\{Q_k\}$ converges to some limiting distribution function G, say. Finally, it is necessary to show that it is possible to construct a random variable Q that has G as its distribution function. In this sense, then, we can say that $Q_k\to Q$. The random variable Q is known as the *steady-state limit* of the sequence of random variables

$\{Q_k\}$, and the distribution G of Q is known as the *limiting* or *stationary distribution* of $\{Q_k\}$.

In these notes we sidestep all these fundamental issues, as the details require measure theory and more advanced probability theory than we can deal with in this course. However, it can all be made precise.

We illustrate the rate of convergence to the limiting situation (i.e., the second question) by means of 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 Eq. (1.16). 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 Eq. (1.16) to compute the *exact* distribution of $W_{Q,k}$ for $k = 1,2,\ldots,20$, c.f., the left panel in Figure 1.10. We see that when $k = 5$, the 'hump' of $\mathsf{P}\left(W_{Q,5} = x\right)$ 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

$$\mathsf{P}\left(W_Q \leq x\right) = \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}[W_{Q,k} \leq x],$$

where $W_{Q,k}$ is obtained by simulation. As should be clear from the figure, the simulated distribution also seems to converge quite fast to some limiting function. Finally, in the right hand panel we compare the densities as obtained by the exact method and simulation with $n = 1000$. Clearly, for all practical purposes, these densities can be treated as the same.

The combination of the fast convergence to the steady-state situation and the difficulties with the transient analysis validates, to some extent, that most queueing theory is concerned with the analysis of the system in *stationarity*. The study of queueing systems in stationary state will occupy us for the rest of the book.

1.6.5. 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.

1.6.6. Validate the results of Figure 1.10 with simulation.

Hints

h.1.6.1 When are the arrivals in slot k available for service in either of the systems?

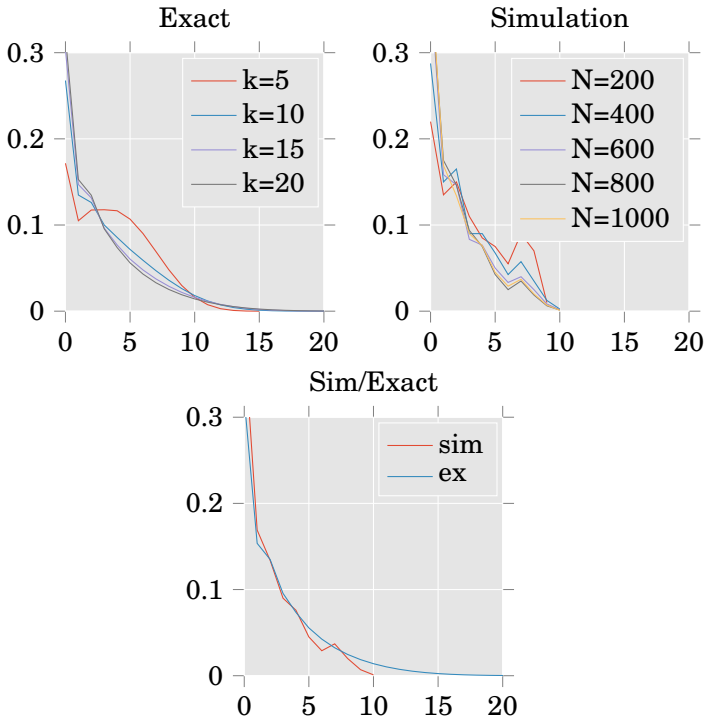


Figure 1.10: The density of $W_{Q,k}$ for $k = 5, 10, 15, 20$ computed by an exact method as co
pared the density obtained by simulation of different run lengths $N = 200, 400, \ldots 10$
The right panel compares the exact density of $W_{Q,20}$ to the density obtained by simulati
for $N = 1000$.

h.1.6.2 Note first that from the expression for Z_k, $a_k - c_k = Z_k - Z_{k-1}$. Use this to get $Q_k = [Q_{k-1} + Z_k - Z_{k-1}]^+$. Subtract Z_k from both sides, use recursion and use subsequently,

$$\begin{aligned}
&\max\{\max\{a,b\},c\} = \max\{a,b,c\},\\
&Q_0 = Z_0,\\
&\max\{-a,-b\} = -\min\{a,b\}.
\end{aligned}$$

h.1.6.4 It is actually not hard, even though the expression looks hard. Use conditioning to see that $\mathsf{P}_m(Z(t)=n) = \mathsf{P}\left(N_\mu(t) - N_\lambda(t) = m-n\right)$. Then write out the definitions of the two Poisson distributions. Assemble terms. Then fiddle a bit with the terms to get $t\sqrt{\lambda\mu}$.

Solutions

s.1.6.1 In the scheme

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

the arrivals are assumed to arrive *at the end* of period k.

When we use the other scheme,

$$\begin{aligned}
Q_k &= [Q_{k-1} + a_k - c_k]^+,\\
d_k &= Q_{k-1} + a_k - Q_k,
\end{aligned}$$

we assume that the arrivals are available at the start of the kth slot. Observe that the number of jobs that enter the kth slot is $Q_{k-1} + a_k$. The number that remain at the end is Q_k. Thus, the number of jobs that depart must be the difference between what enters and what is left behind.

s.1.6.2 Note first that from the expression for Z_k, $a_k - c_k = Z_k - Z_{k-1}$. Using this in the recursion for Q_k, we get

$$Q_k = [Q_{k-1} + Z_k - Z_{k-1}]^+,$$

thus,

$$Q_k - Z_k = \max\{Q_{k-1} - Z_{k-1}, -Z_k\}.$$

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

$$\begin{aligned} Q_k - Z_k &= \max\{\max\{Q_{k-2} - Z_{k-2}, -Z_{k-1}\}, -Z_k\} \\ &= \max\{Q_{k-2} - Z_{k-2}, -Z_{k-1}, -Z_k\} \\ &= \max\{Q_0 - Z_0, -Z_1, \ldots, -Z_k\} \\ &= \max\{0, -Z_1, \ldots, -Z_k\} \\ &= -\min\{0, Z_1, \ldots, Z_k\}. \end{aligned}$$

For further discussion, if you are interested, refer to Baccelli and Massey [1988].

s.1.6.3 From Exercise 1.1.17 we know that thinning a rate λ Poisson process with i.i.d. Bernoulli random variables with success probability p leads to another Poisson process with rate λp. In the present case, the original Poisson process has rate $\lambda+\mu$ and $p = \lambda/(\lambda+\mu)$. Hence, the random variable $N_{\lambda t} \sim P\left(\frac{\lambda}{\lambda+\mu}(\lambda+\mu)t\right) = P(\lambda t)$.

s.1.6.4 With this we have a characterization of the queue length process as a function of time until it hits zero for the first time. What can we say about the distribution of $Q(t)$? With the above random walk,

$$\begin{aligned} \mathsf{P}_m(Z(t)=n) &= \mathsf{P}\left(m + N_\lambda(t) - N_\mu(t) = n\right) = \mathsf{P}\left(N_\lambda(t) - N_\mu(t) = n-m\right) \\ &= \mathsf{P}\left(N_\mu(t) - N_\lambda(t) = m-n\right) \\ &= \sum_{k=0}^{\infty} \mathsf{P}\left(N_\mu(t) = k-n+m \,\middle|\, N_\lambda(t)=k\right)\mathsf{P}(N_\lambda(t)=k) \\ &= \sum_{k=0}^{\infty} e^{-\mu t}\frac{(\mu t)^{k-n+m}}{(k-n+m)!}e^{-\lambda t}\frac{(\lambda t)^k}{k!} \\ &= e^{-(\lambda+\mu)t}\sum_{k=0}^{\infty}\frac{(\lambda t)^k(\mu t)^{k-n+m}}{k!(k-n+m)!}. \end{aligned} \tag{1.27}$$

We can write this a bit simpler by noting that

$$\begin{aligned} (\lambda t)^k(\mu t)^{k+m-n} &= \lambda^k t^k \mu^{k+m-n} t^{k+m-n} \\ &= \lambda^k \mu^{k+m-n}(t\sqrt{\lambda\mu})^{2k+m-n}(\lambda\mu)^{-k+(n-m)/2} \\ &= (\lambda/\mu)^{(n-m)/2}(t\sqrt{\lambda\mu})^{2k+m-n}. \end{aligned}$$

With this,

$$\mathsf{P}(Z(t)=n) = e^{-(\lambda+\mu)t}\left(\frac{\lambda}{\mu}\right)^{(n-m)/2}\sum_{k=0}^{\infty}\frac{(t\sqrt{\lambda\mu})^{2k+m-n}}{k!(k+m-n)!}.$$

s.1.6.5 Here is an example with python. In R it must be equally simple. I compute the difference, i.e., the Kolmogorov-Smirnov statistic, between the distributions of $W_{Q,k-1}$ and $W_{Q,k}$,

$$\max_x\{|\mathsf{P}\left(W_{Q,k} \leq x\right) - \mathsf{P}\left(W_{Q,k-1} \leq x\right)|\},$$

for x in the support of $W_{Q,k}$.

```
>>> from lea import Lea

>>> S = Lea.fromVals(1,  2)
>>> X = Lea.fromVals(1,  3)
>>> U = S-X

>>> W = Lea.fromVals(3)
>>> for k in range(1, 10):
...     W_new = Lea.fastMax(W + U, 0)
...     m = max(abs(W_new.cdf(x)-W.cdf(x))  for x in range(W_new.support
...     print(k, m)
...     W = W_new
...
1 0.5
2 0.1875
3 0.125
4 0.08203125
5 0.0576171875
6 0.042236328125
7 0.03131103515625
8 0.02362060546875
9 0.018096923828125
```

So, after some 10 customers, the distribution hardly changes anymore.

s.1.6.6 You can study `waiting_time_simulation.py` at github if you like, but skipping it is OK. Trying to make the graph in your favorite programming language is fun.

1.7 Rate Stability and Utilization

Theory and Exercises

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, on the long run, jobs arrive faster than they can leave, the system must 'explode'. Thus, the first performance measures we need to estimate when analyzing a queueing system are the arrival and departure rate, and then we need to check that the arrival rate is smaller than the departure rate. In particular, the load, defined as the ratio of the arrival rate and service rate is of importance. In this section we define and relate these concepts. 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.

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.,

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t}. \tag{1.28}$$

We remark in passing that this limit does not necessarily exist if $A(t)$ is some pathological function. If, however, the interarrival times $\{X_k\}$ are the basic data, and $\{X_k\}$ are i.i.d. and distributed as a generic random variable X with finite mean $\mathsf{E}[X]$, we can construct $\{A_k\}$ and $\{A(t)\}$ as described in Section 1.5; the strong law of large numbers guarantees that the above limit exists.

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

Observe that at time $t = A_n$, precisely n arrivals occurred. Thus, by applying the definition of $A(t)$ at the epochs A_n, we see that $A(A_n) = n$. Thus,

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

But since $A_n \to \infty$ if $n \to \infty$, it follows from Eq. (1.28) that the average interarrival time between two consecutive jobs is

$$\mathsf{E}[X] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \lim_{n\to\infty} \frac{A_n}{A(A_n)} = \lim_{t\to\infty} \frac{t}{A(t)} = \frac{1}{\lambda}, \tag{1.29}$$

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 interarrival time.

1.7.2. (This exercise is meant to provide some insight into what needs to be done to put everything on solid ground. If you are not interested in the maths, you can skip the problem.) In Eq. (1.29) we replaced the limit with respect to n by a limit with respect to t. Use the notation $A_{A(t)}$ to show that this is allowed. Show next that the function $t \to A(t)$ as defined by Eqs. (1.15) is right-continuous.

The development of the departure times $\{D_k\}$ is entirely analogous to that of the arrival times; we leave it to the reader to provide the details. As a result we can define the *departure rate* as

$$\delta = \lim_{t\to\infty} \frac{D(t)}{t}. \tag{1.30}$$

1.7.3. Define the departure time D_k of the kth job in terms of $\{D(t)\}$.

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

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

and define the *service* 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) amount of service 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 the number of arrivals, i.e., $D(t) \le A(t)$ for all t. Therefore,

$$\delta := \lim_{t} \frac{D(t)}{t} \le \lim_{t} \frac{A(t)}{t} = \lambda. \tag{1.31}$$

We call a system *(rate) stable* if

$$\lambda = \delta,$$

in other words, the system is stable if, on 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. (It is interesting to prove this.) As it turns out, when $\mu = \lambda$ and the variance of the service times $\mathbb{V}[S] > 0$ or $\mathbb{V}[X] > 0$ then $\lim_t L(t)/t$ does not necessarily exist. From the simulations we also learn that in such cases, the queueing process behaves very peculiar. For this reason we henceforth require that $\mu > \lambda$.

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

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

1.7.6. Show that $\mathsf{E}[S]/\mathsf{E}[X]$ is the fraction of time the server is busy.

1.7.7. Show that $\mathsf{E}[X - S]/\mathsf{E}[X]$ is the fraction of time the server is idle.

The concept of *load* or *utilization*, denoted by the symbol ρ, is fundamental. One way to define this is as the limiting fraction of time the server is busy, i.e.,

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

Interestingly, we can express this in terms of the arrival rate λ and service rate μ. Observe that

$$\sum_{k=1}^{A(t)} S_k \ge \int_0^t \mathbb{1}[L(s) > 0] \, \mathrm{d}s \ge \sum_{k=1}^{D(t)} S_k,$$

since t can lie half way a service interval and $A(t) \geq D(t)$. As $A(t) \to \infty$ as $t \to \infty$,

$$\lim_{t\to\infty} \frac{1}{t} \sum_{k=1}^{A(t)} S_k = \lim_{t\to\infty} \frac{A(t)}{t} \frac{1}{A(t)} \sum_{k=1}^{A(t)} S_k = \lim_{t\to\infty} \frac{A(t)}{t} \cdot \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{A(t)} S_k = \lambda \mathsf{E}[S].$$

Applying similar limits to the other inequality gives

$$\lambda \mathsf{E}[S] \geq \rho \geq \delta \mathsf{E}[S].$$

Hence, if $\delta = \lambda$, $\rho = \lambda \mathsf{E}[S]$.

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., $\mathsf{E}[S] < \mathsf{E}[X]$.

1.7.8. Consider a queueing system with c identical servers (identical in the sense that each server has the same production rate μ). What would be a reasonable stability criterion for this system?

1.7.9. Check with simulation that when $\lambda > \mu$ the queue length grows roughly linearly with slope $\lambda - \mu$. Thus, if $\rho > 1$, 'we are in trouble'.

1.7.10. If $\mathsf{E}[B]$ is the expected busy time and $\mathsf{E}[I]$ is the expected idle time, show that $\mathsf{E}[B] = \mathsf{E}[I]\rho/(1-\rho)$ is the fraction of time the server is busy.

1.7.11. Consider a paint factory which 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_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:

$$\text{minimize } T$$

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

Hints

h.1.7.1 As a start, the function $\sin(t)$ has not a limit as $t \to \infty$. However, the time-average $\sin(t)/t \to 0$. Now you need to make some function whose time-average does not converge, hence it should grow fast, or fluctuate wilder and wilder.

h.1.7.2 Use that $A_{A(t)} \le t < A_{A(t)+1}$. Divide by $A(t)$ and take suitable limits. BTW, such type of proof is used quite often to show that the existence of one limit implies, and is implied by, the existence of another type of limit.

h.1.7.3 Use the analogy with Eq. (1.13).

h.1.7.5 Remember that $\{X_k\}$ and $\{S_k\}$ are sequences of i.i.d. random variables. What are the implications for the expectations?

h.1.7.6 Suppose that the $n+1$th job is the first job after time A_1 that sees an empty system. Thus, job n leaves an empty system behind. The very first job that arrives, at time $A_1 = X_1$, also sees an empty system. The total amount of service that arrived during $[A_1, A_{n+1})$ is $\sum_{i=1}^{n} S_i$. What is the fraction of time the server has been busy during $[A_1, A_{n+1})$?

h.1.7.8 What is the rate in, and what is the service capacity?

h.1.7.11 Here are some questions to help you interpret this formulation.

1. What are the decision variables for this problem? In other words, what are the 'things' we can control/change?
2. What are the interpretations of $k_A t_A$, and $S + k_B t_B$?
3. What is the meaning of the first constraint? Realize that T represents one production cycle. After the completion of one such cycle, we start another cycle. Hence, the start of every cycle can be seen as a restart of the entire system.

4. What is the meaning of the other two constraints?

5. Why do we minimize the cycle time T?

6. Solve for k_A and k_B in terms of S, λ_A, λ_B and t_A, t_B.

7. Generalize this to m job classes and such that the cleaning time between jobs of class i and j is given by S_{ij}. (Thus, the setup times are sequence dependent.)

Solutions

s.1.7.1 If $N(t) = 3t^2$, then clearly $N(t)/t = 3t$. This does not converge to a limit.

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

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

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

For the mathematically interested, we seek a function for which its Cesàro limit does not exist.

s.1.7.2 Observing that $A_{A(t)}$ is the arrival time of the last job before time t and that $A_{A(t)+1}$ is the arrival time of the first job after time t:

$$A_{A(t)} \le t < A_{A(t)+1} \Leftrightarrow \frac{A_{A(t)}}{A(t)} \le \frac{t}{A(t)} < \frac{A_{A(t)+1}}{A(t)} = \frac{A_{A(t)+1}}{A(t)+1} \frac{A(t)+1}{A(t)}$$

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

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

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

We need to prove, for right-continuity, that $f(u) \to f(t)$ as $u \downarrow t$. When $f(t) = 1$, $f(u) = 1$ for any $u > 1$, by the definition of $f(x)$. When $f(t) = 0$ we have to do a bit more work. Formally, we have to prove that, for fixed t and for all $\epsilon > 0$, there is a

$\delta > 0$ such that $u \in (t, t+\delta) \Rightarrow |f(u) - f(t)| < \epsilon$. (Note the differences with the regular definition of continuity.) Since, by assumption, t is such that $f(t) = 0$, and $f \in \{0,1\}$ we need to show that $f(u) = 0$ for $u \in (t, t+\delta)$. Now, clearly, $f(t) = 0$ only if $t < A_1$. But, then for any $u \in (t, A_1)$, we have that $f(u) = 0$. Thus, taking $\delta = A_1 - t$ suffices.

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

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

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

s.1.7.3

$$D_k = \inf\{t; D(t) \geq k\}.$$

s.1.7.4 That the average time customers spend in service is smaller than the average time between the arrival of two subsequent jobs.

s.1.7.5 $0 > \mathsf{E}\left[\tilde{X}_k\right] = \mathsf{E}[S_{k-1} - X_k] = \mathsf{E}[S_{k-1}] - \mathsf{E}[X_k] = \mathsf{E}[S] - \mathsf{E}[X]$, where we use the fact that the $\{S_k\}$ and $\{X_k\}$ are i.i.d. sequences. Hence,

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

s.1.7.6 The fraction of time that the server has been busy during $[A_1, A_{n+1})$ is

$$\frac{\sum_{i=1}^{n} S_i}{A_{n+1} - A_1} = \frac{\sum_{i=1}^{n} S_i}{\sum_{i=1}^{n+1} X_i - X_1} = \frac{\sum_{i=1}^{n} S_i}{\sum_{i=2}^{n+1} X_i}$$

Now use the assumption that the $\{X_i\}$ and $\{S_i\}$ are sequences of i.i.d. random variables distributed as the generic random variables X and S, respectively. Then by taking expectations the above becomes

$$\frac{\mathsf{E}\left[\sum_{i=1}^{n} S_i\right]}{\mathsf{E}\left[\sum_{i=2}^{n+1} X_i\right]} = \frac{n\,\mathsf{E}[S]}{n\,\mathsf{E}[X]} = \frac{\mathsf{E}[S]}{\mathsf{E}[X]}.$$

We call periods like $[A_1, A_{n+1})$ 'busy cycles', as they start with the moment the server start working, until the next arrival after the server has been idle.

Such busy cycles occur over and over again. Thus, the long-run average fraction of time the server is busy must also be $\mathsf{E}[S]/\mathsf{E}[X]$. (For the serious student, there is a subtle point here: the arrival epochs of the $G/G/1$ queue are not real renewal moments, hence the epochs at which the busy times start also do not form a sequence of renewal times. But then it is not true, in general, that the busy times $\{B_i\}$ have the same distribution, neither do the idles times $\{I_n\}$. Showing that in the limit all is OK requires a substantial amount of mathematics. The above claim is still true however.)

s.1.7.7 Since the fraction of idle time is 1 minus the fraction of busy time, it follows that $1 - \mathsf{E}[S]/\mathsf{E}[X] = (\mathsf{E}[X] - \mathsf{E}[S])/\mathsf{E}[X]$ is the idle time fraction.

s.1.7.8 The criterion is that c must be such that $\lambda < c\mu$. (Thus, we interpret the number of servers as a *control*, i.e., a 'thing' we can change, while we assume that λ and μ cannot be easily changed.) To see this, we can take two different points of view. Imagine that the c servers are replaced by one server that works c times as fast. The service capacity of these two systems (i.e., the system with c servers and the system with one fast server) is the same, i.e., $c\mu$, where μ is the rate of one server. For the system with the fast server the load is defined as $\rho = \lambda/c\mu$, and for stability we require $\rho < 1$. Another way to see it is to assume that the stream of jobs is split into c smaller streams, each with arrival rate λ/c. In this case, applying the condition that $(\lambda/c)/\mu < 1$ per server leads to the same condition that $\lambda/(c\mu) < 1$.

s.1.7.9 See my website.

s.1.7.10 Consider a busy cycle, that is, a cycle that starts with the first job that sees an empty system at upon arrival up to the time another job sees an empty system upon arrival. In such one cycle, the server is busy for an expected duration $\mathsf{E}[B]$. The total expected length of the cycle is $\mathsf{E}[B] + \mathsf{E}[I]$, since after the last job of the cycle left, the expected time until the next job is $\mathsf{E}[I]$.

Since ρ is utilization of the server,

$$\rho = \frac{\mathsf{E}[B]}{\mathsf{E}[B] + \mathsf{E}[I]}.$$

With a bit of algebra the result follows.

s.1.7.11 Realize that the machine works in cycles. A cycle starts with processing k_A jobs of type A, then does a setup, and processes k_B jobs of type B, and then a new cycle starts again. The time it takes to complete one such cycle is $T = k_A t_A + S +$

$k_B t_B$. The number of jobs of type A processed during one such cycle is, of course, k_A. Observe next that the average number of jobs that arrive during one cycle is $\lambda_A T$. We of course want that $\lambda_A T < k_A$, i.e., less jobs of type A arrive on average per cycle than what we can process.

1.8 (Limits of) Empirical Performance Measures

Theory and Exercises

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; the most important being the utilization ρ. We provide an overview of the relations between these performance measures in Figure 1.11.

With the construction of queueing processes in Section 1.5 we can compute the waiting time as observed by the first n, say, jobs. Thus, the average waiting time of the first n arrivals is given by $n^{-1}\sum_{k=1}^{n} W_k$. We therefore define the *expected waiting time* as

$$\mathsf{E}[W] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} W_k, \tag{1.32}$$

and the expected time in queue as

$$\mathsf{E}\left[W_Q\right] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} W_{Q,k}. \tag{1.33}$$

Note that these performance measures are limits of *emperical* 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:

$$\mathsf{P}(W \le x) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}[W_k \le x]. \tag{1.34}$$

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

$$\mathsf{E}[L] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} L(A_k-), \tag{1.35}$$

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

$$\mathsf{P}(L \le m) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}[L(A_k-) \le m]. \tag{1.36}$$

1.8.1. Assume that $X_k = 10$ minutes and $S_k = 11$ minutes for all k, i.e., X_k and S_k are deterministic and constant. What are λ and μ? Compute A_k, W_k, D_k and $L(A_k-)$. How do these numbers behave as a function of k?

1.8.2. Yet another simple case is to take $X_k = 10$ minutes and $S_k = 9$ minutes for all k. Answer the same questions as in the previous exercise.

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. Thus, if we simulate the queueing system up to time t, the *time-average number of jobs in the system* is given by

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \frac{1}{t}\int_0^t (A(s) - D(s))\,\mathrm{d}s, \tag{1.37}$$

where we use that $L(t) = A(t) - D(t) + L(0)$ is the total number of jobs in the system at time t and $L(0) = 0$, c.f. Figure 1.7. Observe from the second equation that $\int_0^t L(s)\,\mathrm{d}s$ is the area enclosed between the graphs of $\{A(t)\}$ and $\{D(t)\}$. Assuming the limit exists for $t \to \infty$, we define

$$\mathsf{E}[L] = \lim_{t\to\infty} \frac{1}{t}\int_0^t L(s)\,\mathrm{d}s \tag{1.38}$$

Observe that, notwithstanding that the symbols are the same, this expectation need not be the same as (1.35): in general,

$$\lim_{t\to\infty} \frac{1}{t}\int_0^t L(s)\,\mathrm{d}s \neq \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} L(A_k-).$$

1.8.3. Design a queueing system to show that average number of jobs in the system as seen by the server is very different from what the customers see.

Next, define the following probability as the *time-average fraction of time the system contains at most m jobs*:

$$\mathsf{P}(L \le m) = \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathbb{1}[L(s) \le m]\,\mathrm{d}s. \tag{1.39}$$

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

1.8.4. Formulate a definition for the time-average of the waiting time.

1.8.5. Consider a discrete-time model of a queueing system, as we developed in Section 1.4, which case batches of jobs arrive in a period. The average number of customers that *see upon arrival* more than m customers in the system cannot be defined as (1.36). Provide a better definition.

Hints

h.1.8.3 Consider a queueing system with constant service and interarrival times.

h.1.8.5 Why is (1.36) not the same as the number of batches that see a queue length less than m?

Solutions

s.1.8.1 $\lambda = 6$ per hour, and $\mu = 60/11$ per hour. Note that $\mu < \lambda$. $A_0 = 0$, $A_1 = 10$, $A_2 = 20$, etc., hence $A_k = 10k$. $W_{Q,0} = 0$, $W_{Q,1} = \max\{0 + 0 - 10, 0\} = 0$. $W_{Q,2} = \max\{0 + 11 - 10, 0\} = 1$. $W_{Q,3} = \max\{1 + 11 - 10, 0\} = 2$. Hence, $W_{Q,k} = k - 1$ for $k \ge 1$. Thus, $W_k = k - 1 + 11 = k + 10$ for $k \ge 1$, and $D_k = 10k + k + 10 = 11k + 10$. Note that W_k increases linearly as a function of k. You can infer that, for the queue length to remain bounded, it is necessary that the service rate $\mu > \lambda$.

s.1.8.2 Trivial.

s.1.8.3 Take $X_k = 10$ and $S_k = 10 - \epsilon$ for some tiny ϵ. Then $L(t) = 1$ nearly all of the time. In fact, $\mathsf{E}[L] = 1 - \epsilon/10$. However, $L(A_k-) = 0$ for all k.

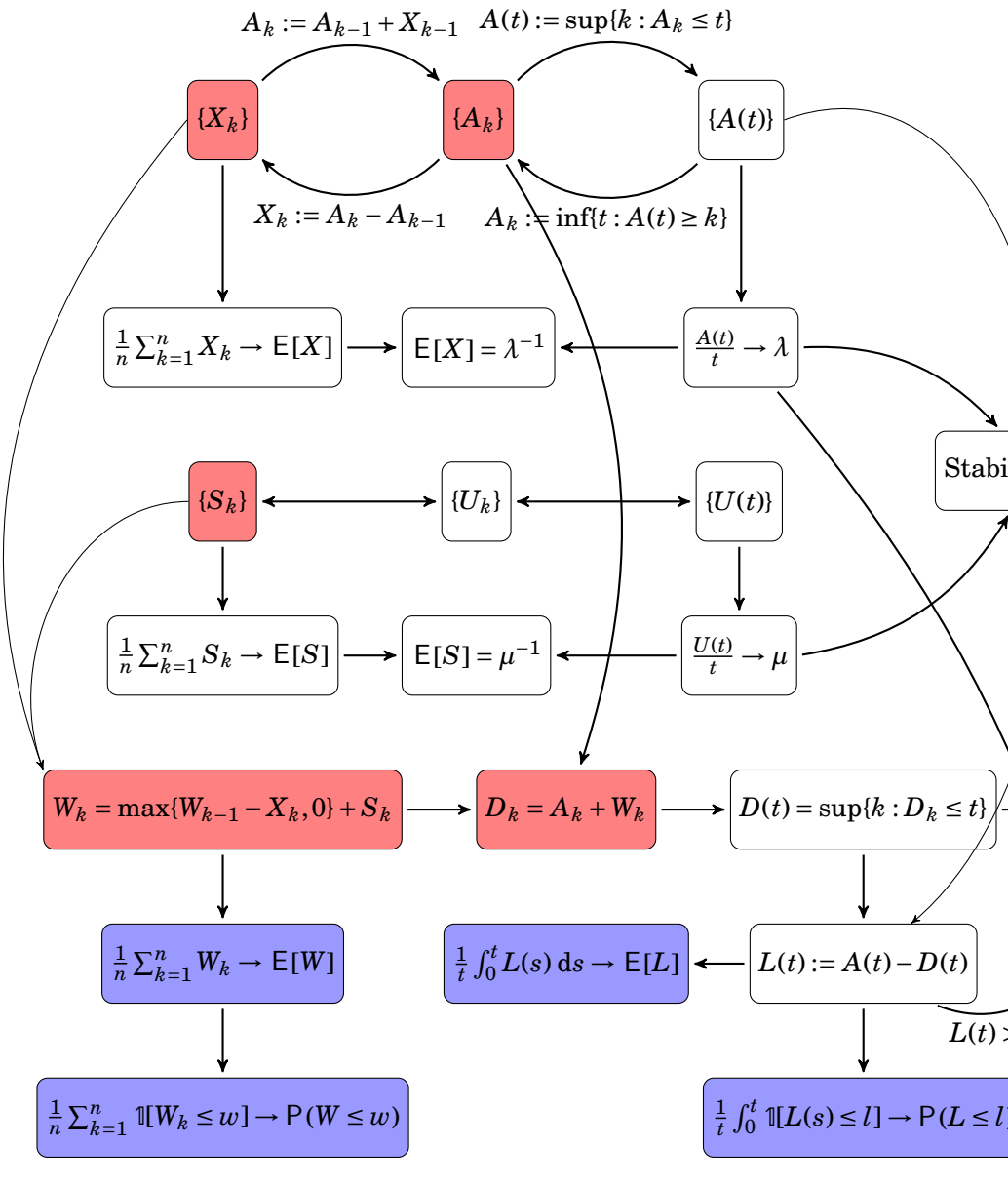


Figure 1.11: Here we sketch the relations between the construction of the $G/G/1$ que
from the primary data, i.e., the interarrival times $\{X_k; k \ge 0\}$ and the service times $\{S_k;$
0}, and different performance measure.

s.1.8.4

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

s.1.8.5 Since we deal with a system in discrete time, L_k is the queue length at the end of period k. Thus, $\sum_{k=1}^{n} \mathbb{1}[L_k > m]$ counts the number of *periods* that the queue is larger than m. This is of course not the same as the number of *jobs* that see a queue larger than m; only when $a_k > 0$ the customers in a batch would see a queue $L_k > m$. Thus,

$$\sum_{k=1}^{n} \mathbb{1}[L_k > m]\,\mathbb{1}[a_k > 0],$$

counts the number of batches.

Next, by assumption, a_k customers arrive during period k. The first of these customers sees a queue length of $L_{k-1} - d_k$, the second $L_{k-1} - d_k + 1$, and so on until the last customer who sees a queue length of $L_{k-1} - d_k + a_k - 1 = L_k - 1$. Thus, of all jobs the last sees the largest queue. Hence, if $L_k \le m$, all customers of the batch see a queue less than m. If, however, $L_k > m$, then $L_k - m$ customers saw m or more jobs in the system. Therefore, the fraction of arrivals that see a queue with m or more jobs is equal to

$$\frac{1}{A(n)} \sum_{k=1}^{n} (L_k - m)\,\mathbb{1}[L_k > m].$$

We could also define this a bit differently. Suppose that we don't want to distinguish between jobs in a batch, but simply want to say that if one job sees a long queue, all see a long queue. In that case,

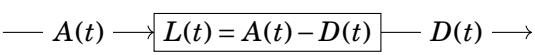
$$\frac{1}{A(n)} \sum_{k=1}^{n} a_k\,\mathbb{1}[L_k > m].$$

Thus, when jobs arrive in batches, the definition of loss fraction requires some care; not all definitions need to measure the same.

1.9 Level-Crossing and Balance Equations

Theory and Exercises

Consider a system at which customers arrive and depart in single entities, such as customers in a shop or jobs at some machine. If the system starts empty, then we know that the number $L(t)$ is the system at time t is equal to $A(t) - D(t)$. To illustrate:



What goes in the box (i.e., $A(t)$) and has not yet left (i.e., $D(t)$) must be in the box, hence $L(t) = A(t) - D(t)$.

Let us denote an arrival as an 'up-crossing' and a departure as a 'down-crossing'. Then, clearly $L(t)$ is the number of up-crossings up to time t minus the number of down-crossings up to time t. If $L(t)$ remains finite, or more generally $L(t)/t \to 0$ as $t \to \infty$, then it must be that

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

Hence, when $L(t)/t \to 0$, the *up crossing rate* $\lim_t A(t)/t = \lambda$ is equal to the *down-crossing rate* $\lim_t D(t)/t = \delta$. We will generalize these notions of up- and downcrossing in this section to derive the *stationary*—also known as *long-run time average* or *steady-state*, distribution—$p(n)$ that the system contains n jobs.

1.9.1. If $L(t)/t \to 0$ as $t \to \infty$, can it still be true that $\mathsf{E}[L] > 0$?

Let us say that the system is in *state n when it contains n jobs*, c.f. Figure 1.12, and the system *crosses* level n when the state changes from n to $n+1$, either 'from below' when an arrival occurs, or 'from above' when a departure occurs.

The main result we derive in this section is that the rate at which level n is crossed from below must match the rate at which it is crossed from above, i.e.,

$$\lambda(n)p(n) = \mu(n+1)p(n+1),$$

where $\lambda(n)$ is the arrival rate in state n, $p(n)$ the fraction of time the system is in state n and $\mu(n+1)$ the departure rate from $n+1$.

To establish this 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. Once we have the proper definitions, the above result will follow straightaway.

Define[2]

$$A(n,t) = \sum_{k=1}^{\infty} \mathbb{1}[A_k \le t]\,\mathbb{1}[L(A_k-) = n] \tag{1.40a}$$

as the number of arrivals up to time t that saw n customers in the system at their arrival.

[2] It is common to write $f(x+) = \lim_{h \downarrow 0} f(x+h)$ for the right limit of a function f at x and $f(x-) = \lim_{h \downarrow 0} f(x-h)$ for the left limit. When $f(x-) = f(x+)$, f is continuous at x. Since in queueing systems we are concerned with processes with jumps, we need to be quite particular about left and right limits at jump epochs.

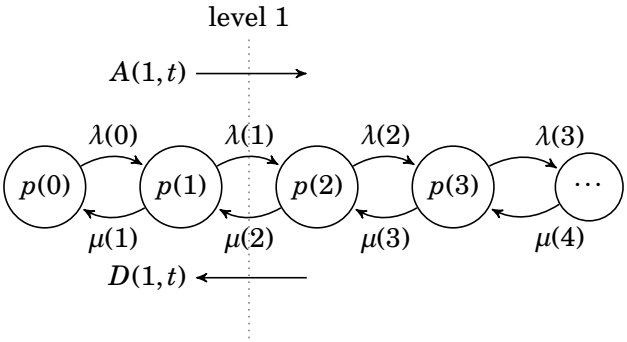


Figure 1.12: $A(1,t)$ counts the number of jobs up to time t that saw 1 job in the syste thus, right after the arrival of such a job the system contains 2 jobs. Similarly, $D(1$ counts the number of departures that leave 1 job behind. Each time $A(1,t)$ increases one or $D(1,t)$ decreases by one, level 1 (the dotted line separating states 1 and 2) is cross The number of times this level is crossed from below must be the same (plus or minus the number of times it is crossed from above.

1.9.2. Should we take $L(A_k-) = n$ or $L(A_k)$ in the definition of $A(n,t)$?

1.9.3. What is the difference between $A(n,t)$ and $A(t)$?

1.9.4. Show that $A(n,t) \le A(t)$.

1.9.5. Why is $\lim_{t\to\infty} \frac{A(n,t)}{t} = 0$ if $\lambda > \delta$, i.e., if the system is not rate-stable?

1.9.6. 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. What acronym would describe this queueing situation? Find an expression for $A(n,t)$.

Next, let

$$Y(n,t) = \int_0^t \mathbb{1}[L(s) = n]\,\mathrm{d}s \tag{1.40b}$$

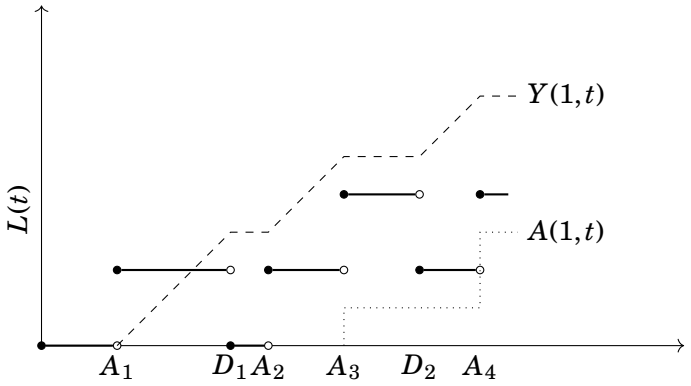


Figure 1.13: Plots of $Y(1,t)$ and $A(1,t)$. (For visual clarity, we subtracted ½ from $A(1$ for otherwise its graph would partly overlap with the graph of L.)

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

$$p(n,t) = \frac{1}{t}\int_0^t \mathbb{1}[L(s)=n]\,\mathrm{d}s = \frac{Y(n,t)}{t}, \tag{1.40c}$$

be the fraction of time that $L(s) = n$ in $[0,t]$. Figure 1.13 illustrates the relation between $Y(n,t)$ and $A(n,t)$.

1.9.7. Continuation of Exercise 1.9.6. Find an expression for $Y(n,t)$.

Define also the limits:

$$\lambda(n) = \lim_{t\to\infty} \frac{A(n,t)}{Y(n,t)}, \qquad p(n) = \lim_{t\to\infty} p(n,t), \tag{1.41}$$

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 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 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 divided by the time the system contains n jobs.

1.9.8. Continuation of Exercises 1.9.6 and 1.9.7. Compute $p(n)$ and $\lambda(n)$.

Similar as the definition for $A(n,t)$, denote by

$$D(n,t) = \sum_k \mathbb{1}[D_k \le t]\,\mathbb{1}[L(D_k) = n]$$

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

1.9.9. Should we take $D(n-1,t)$ or $D(n,t)$ in the definition of $\mu(n)$?

1.9.10. Continuation of Exercises 1.9.6, 1.9.7, and 1.9.8. 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

$$\lim_{t\to\infty} \frac{A(n,t)}{t} = \lim_{t\to\infty} \frac{D(n,t)}{t}. \tag{1.42}$$

With this equation we can obtain two nice and fundamental identities. The first we develop now; the second follows in Section 1.12.

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

$$\frac{A(n,t)}{t} = \frac{A(n,t)}{Y(n,t)} \frac{Y(n,t)}{t} \to \lambda(n)p(n), \tag{1.43a}$$

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

$$\frac{D(n,t)}{t} = \frac{D(n,t)}{Y(n+1,t)} \frac{Y(n+1,t)}{t} \to \mu(n+1)p(n+1). \tag{1.43b}$$

Combining this with (1.42) we arrive at *the level-crossing equations*

$$\lambda(n)p(n) = \mu(n+1)p(n+1). \tag{1.44}$$

1.9.11. Continuation of Exercises 1.9.6, 1.9.7, and 1.9.8. Compute $\lambda(n)p(n)$ for $n \geq 0$, and check $\lambda(n)p(n) = \mu(n+1)p(n+1)$.

The result 1.44 turns out to be exceedingly useful, as will become evident from Section 1.10 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

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)} p(n). \tag{1.45}$$

Thus, if we have $p(n)$ we can compute $p(n+1)$, and so on. In other words, if $p(0)$ is known, than $p(1)$ follows, from which $p(2)$ follows, and so on. A straightaway iteration then leads to

$$p(n+1) = \frac{\lambda(n)\lambda(n-1)\cdots\lambda(0)}{\mu(n+1)\mu(n)\cdots\mu(1)} p(0). \tag{1.46}$$

Finally, to determine $p(0)$ we can use the fact that numbers $p(n)$ have to be normalized. Let the *normalization constant* be given by

$$G = 1 + \frac{\lambda(0)}{\mu(1)} + \frac{\lambda(0)\lambda(1)}{\mu(1)\mu(2)} + \cdots. \tag{1.47}$$

Then, $p(0) = G^{-1}$ and $p(n)$ follows from Eq. (1.46).

Let us now express a few important performance measures in terms of $p(n)$: the average number of items $\mathsf{E}[L]$ in the system and the fraction of time $\mathsf{P}(L \leq n)$ the system contains at least n jobs. As $L(s)$ counts the number of jobs in the system at time s (thus $L(s)$ is an integer),

$$L(s) = \sum_{n=0}^{\infty} n \, \mathbb{1}[L(s) = n].$$

With this we can write for the time-average number of jobs in the system

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \frac{1}{t}\int_0^t \left(\sum_n n \, \mathbb{1}[L(s) = n]\right) \mathrm{d}s = \sum_n \frac{n}{t}\int_0^t \mathbb{1}[L(s) = n]\,\mathrm{d}s, \tag{1.48}$$

where we interchange the integral and the summation[3]. It then follows from Eq. (1.40c) that

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \sum_n n\, p(n,t).$$

[3] This is allowed as the integrand is non-negative. More generally, the interested reader should check Fubini's theorem.

Finally, assuming that the limit $p(n,t) \to p(n)$ exists as $t \to \infty$ (and that the summation and limit can be interchanged in the above), it follows that

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s \to \sum_{n=0}^{\infty} np(n) = \mathsf{E}[L], \text{ as } t \to \infty.$$

In a loose sense we can say that $\mathsf{E}[L]$ is the average number in the system as perceived by the *server*. (Recall that this is not necessarily the same as what *arriving* jobs 'see'). Similarly, the probability that the system contains at least n jobs is

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

From the above we conclude that from the probabilities $p(n)$, more specifically, from a specification of $\lambda(n)$ and $\mu(n)$, we can compute numerous performance measures. Thus, determining $p(n)$ from Eqs. (1.45) and (1.47) for concrete queueing examples is the task of the next sections.

Before we embark on this task, we note that the level-crossing cannot always be used as we do here. The reason is that it is not always natural, or easy, to decompose the state space 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, c.f. Figure 1.14. 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 or a departure to state $n-1$.

Of course, $|I(n,t) - O(n,t)| \leq 1$. Thus, from the fact that

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

and

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

we get that

$$\lambda(n-1)p(n-1) + \mu(n+1)p(n+1) = (\lambda(n) + \mu(n))p(n).$$

These equations hold for any $n \geq 0$ and are known as the *balance equations*. We will use these equations when studying queueing systems in which level crossing cannot be used, for instance for queueing networks.

Again, just by using properties, i.e., counting differences, that hold along any sensible sample path we obtain very useful statistical and probabilistic results.

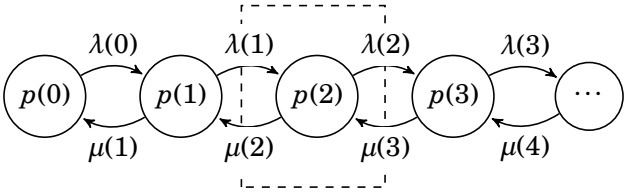


Figure 1.14: For the balance equations we count how often a box around a state is cross from inside and outside. On the long run the entering and leaving rates should be equ For the example here, the rate out is $p(2)\lambda(2) + p(2)\mu(2)$ while the rate in is $p(1)\lambda(1$ $p(3)\mu(3)$.

Interpretation The definitions in (1.40) 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 Eq. (1.40a) is 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 want 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 is $wnY(n,t)$ the total cost up to time t to have n

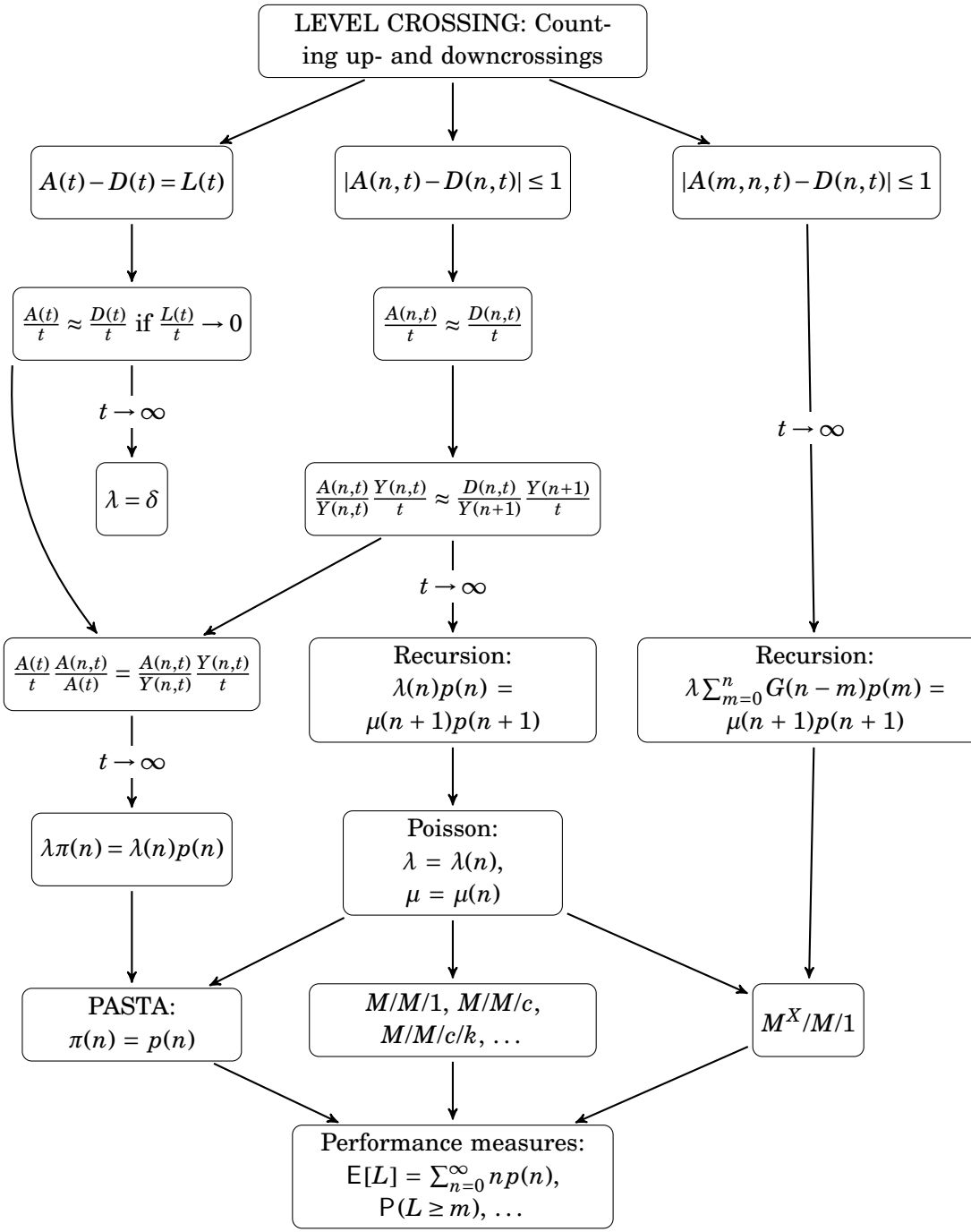


Figure 1.15: With level-crossing arguments we can derive a number of useful relatio
This figure presents an overview of these relations that we derive in this and the n
sections.

jobs in queue, hence the total cost up up to time t

$$C(t) = w \sum_{n=0}^{\infty} nY(n,t),$$

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.

Finally, the following two exercises show that level-crossing arguments extend well beyond the queueing systems modeled by Figure 1.12.

1.9.12. 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 interarrival 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.

1.9.13. Use the graph of the previous question and a level-crossing argument to express the steady-state probabilities $p(n), n = 0, \ldots, 3$ in terms of λ and μ.

Hints

h.1.9.2 Recall that $L(t)$ is *right-continuous*.

h.1.9.6 For the acronym, observe that the service times and interarrival are *D*eterminis and there is one server. For the computation of $Y(n,t)$, make a plot of $L(s)$ as a function of time for $n = 1$.

Make a plot of $L(s)$ for $n = 1$ as a function of time.

h.1.9.13 First balance the rates accross the levels. Then solve for $\pi(k)$.

Solutions

s.1.9.1

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

If $L(t) = 1$ for all t, $\mathsf{E}[L] = 1$, but $L(t)/t \to 0$.

s.1.9.2 $L(t)$ is the number of customers in the system at time t. As such the function $t \to L(t)$ is *right-continuous.* The definition of $L(A_k-) = \lim_{t\uparrow A_k} L(t)$ is the limit from the left. The customer therefore 'sees' $L(A_k-)$ just before he arrives.

s.1.9.3 $A(t)$ counts all customers that arrive up to time t, i.e., during $[0,t]$. Note that this *includes* time t. $A(n,t)$ counts the jobs that see n jobs in the system just before they arrive.

s.1.9.4 Observe that $\mathbb{1}[A_k \le t]\,\mathbb{1}[L(A_k-) = n] \le \mathbb{1}[A_k \le t]$; the last inequality follows from the fact that $\mathbb{1}[L(A_k-) = n] \le 1$. Therefore,

$$A(n,t) = \sum_{k=1}^{\infty} \mathbb{1}[A_k \le t]\,\mathbb{1}[L(A_k-) = n] \le \sum_{k=1}^{\infty} \mathbb{1}[A_k \le t] = A(t).$$

For any 'normal' queueing system, $A(t) > A(t,n)$, because the queue length fluctuates.

s.1.9.5 If $\lambda > \delta$, then $L(t) \to \infty$. But then there must be last time, s say, that $L(s) = n+1$, and $L(t) > n+1$ for all $t > s$. Hence, after time s no job will see the system with n jobs. Thus $A(t,n) = A(s,n)$ for all $t > s$. This is a finite number, while $t \to \infty$, so that $A(n,t)/t \to 0$.

s.1.9.6 It is the $D/D/1$ queue, since there is one server and the interarrival times and service times are constant, i.e., *D*eterministic.

$A_k = 2k$ as jobs arrive at $t = 0,2,4,\ldots$, hence, $A(t) \approx t/2$ when $t \gg 0$. We also know that $L(s) = 1$ if $s \in [2i, 2i+1)$ and $L(s) = 0$ for $s \in [2i-1, 2i)$ for $i = 0,1,2,\ldots$. Thus, $L(A_k-) = L(2k-) = 0$. Hence, $A(0,t) \approx t/2$ for $t \gg 0$, and $A(n,t) = 0$ for $n \ge 1$.

s.1.9.7 Next, to get $Y(n,t)$, observe that the system never contains more than 1 job. Hence, $Y(n,t) = 0$ for all $n \ge 2$. Then we see that $Y(1,t) = \int_0^t \mathbb{1}[L(s) = 1]\,\mathrm{d}s$. Now observe that for our queueing system $L(s) = 1$ for $s \in [0,1)$, $L(s) = 0$ for $s \in [1,2)$, $L(s) = 1$ for $s \in [2,3)$, and so on, Thus, when $t < 1$, $Y(1,t) = \int_0^t \mathbb{1}[L(s) = 1]\,\mathrm{d}s = \int_0^t 1\,\mathrm{d}s = t$. When $t \in [1,2)$,

$$L(t) = 0 \implies \mathbb{1}[L(t) = 0] \implies Y(1,t) \text{ does not change.}$$

Continuing to $[2,3)$ and so on gives

$$Y(1,t)=\begin{cases} t & \le t\in[0,1),\\ 1 & \le t\in[1,2),\\ 1+(t-2) & \le t\in[2,3),\\ 2 & \le t\in[3,4),\\ 2+(t-4) & \le t\in[4,5),\end{cases}$$

and so on. Since $Y(n,t)=0$ for all $n\ge 2$, $L(s)=1$ or $L(s)=0$ for all s, therefore,

$$Y(0,t)=t-Y(1,t).$$

s.1.9.8 From the other exercises:

$$\begin{aligned}\lambda(0)&\approx\frac{A(0,t)}{Y(0,t)}\approx\frac{t/2}{t/2}=1,\\ \lambda(1)&\approx\frac{A(1,t)}{Y(1,t)}\approx\frac{0}{t/2}=0,\\ p(0)&\approx\frac{Y(0,t)}{t}\approx\frac{t/2}{t}=\frac{1}{2},\\ p(1)&\approx\frac{Y(1,t)}{t}\approx\frac{t/2}{t}=\frac{1}{2}.\end{aligned}$$

For the rest $\lambda(n)=0$, and $p(n)=0$, for $n\ge 2$.

s.1.9.9 $D(n-1,t)$ counts the departures that leave $n-1$ behind. Thus, just before the customer leaves, the system contains n customers.

s.1.9.10 $D(0,t)=\sum_{k=1}^{\infty}\mathbb{1}[D_k\le t, L(D_k)=0]$. From the graph of $\{L(s)\}$ we see that all jobs leave an empty system behind. Thus, $D(0,t)\approx t/2$, and $D(n,t)=0$ for $n\ge 1$. With this, $D(0,t)/Y(1,t)\sim(t/2)/(t/2)=1$, and so,

$$\mu(1)=\lim_t\frac{D(0,t)}{Y(1,t)}=1,$$

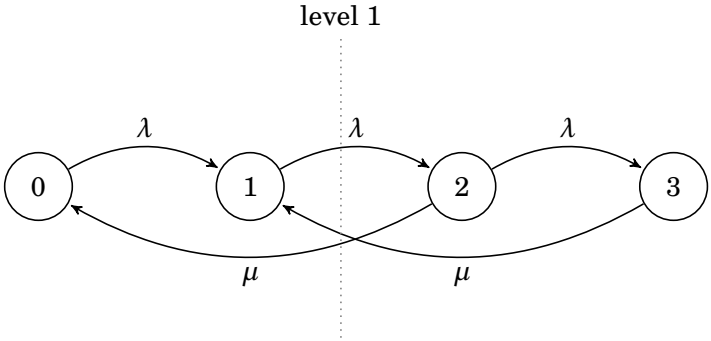
and $\mu(n)=0$ for $n\ge 2$.

s.1.9.11 $\lambda(0)p(0)=1\cdot 1/2=1/2$, $\lambda(n)p(n)=0$ for $n>1$, as $\lambda(n)=0$ for $n>0$.

From Exercise 1.9.10, $\mu(1)=1$, hence $\mu(1)p(1)=1\cdot 1/2=1/2$. Moreover, $\mu(n)=0$ for $n\ge 2$.

Clearly, for all n we have $\lambda(n)p(n)=\mu(n+1)p(n+1)$.

s.1.9.12 See the figure below.



s.1.9.13 With level crossing

$$\begin{aligned}
\lambda p(0) &= \mu p(2) \quad \text{the level between 0 and 1} \\
\lambda p(1) &= \mu p(2) + \mu p(3), \quad \text{see level 1} \\
\lambda p(2) &= \mu p(3) \quad \text{the level between 2 and 3}
\end{aligned}$$

Solving this in terms of $p(0)$ gives $p(2) = \rho p(0)$, $p(3) = \rho p(2) = \rho^2 p(0)$, and

$$\lambda p(1) = \mu(p(2) + p(3)) = \mu(\rho + \rho^2)p(0) = (\lambda + \lambda^2/\mu)p(0),$$

hence $p(1) = p(0)(\mu + \lambda)/\mu$.

1.10 M/M/1 queue

Theory and Exercises

In the $M/M/1$ queue, one server serves jobs arriving with exponentially distributed interarrival times and each job requires an exponentially distributed processing time. With Eq. (1.45), i.e., $\lambda(n)p(n) = \mu(n+1)p(n+1)$ we can derive a number of important results for this queueing process.

Recall from Section 1.6 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 in the $L(t) > 0$) 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, or the queue for that matter, $\lambda(n) = \lambda$ and $\mu(n) = \mu$ for all n. Thus, (1.45) 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, from the normalization condition

$$1 = \sum_{n=0}^{\infty} p(n) = p(0) \sum_{n=0}^{\infty} \rho^n = \frac{p(0)}{1-\rho},$$

it follows that

$$p(0) = 1 - \rho, \qquad p(n) = (1-\rho)\rho^n. \tag{1.49}$$

How can we use these equations? First, note that $p(0)$ must be the fraction of time the server is idle. Hence, the fraction of time the server is busy, i.e., the utilization, is

$$1 - p(0) = \rho = \sum_{n=1}^{\infty} p(n).$$

Here the last equation has the interpretation of the fraction of time the system contains at least 1 job.

1.10.1. Show that the average number of jobs in an $M/M/1$ queue is given by

$$\mathsf{E}[L] = \frac{\rho}{1-\rho}. \tag{1.50}$$

From the previous exercise, it follows that

$$\mathsf{E}[L] \sim \frac{1}{1-\rho}, \quad \text{as } \rho \to 1..$$

Let us interpret this expression. 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.

1.10.2. Show that the excess probability, i.e., the probability that a long queue occurs, is $\mathsf{P}(L \geq n) = \rho^n$.

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 of the fraction of customers lost.

The aim of the next set of exercises is to determine the distribution of the inter-departure times of the $M/M/1$ queue. We will need these results when we analyze networks of queues—observe that in a network of queues the departures from one queueing station form the arrivals at another station. We chop up this problem in small steps to help you find the answer mostly by yourself.

1.10.3. Why is the output rate of the (stable) $M/M/1$ queue equal to λ and not μ?

1.10.4. Why is $\mu e^{-\mu t}$ not a reasonable density for the inter-departure times?

The simplest guess for the inter-departure density might be $\lambda e^{-\lambda t}$; so this is what we will try to prove. As we will see, this result hold.

1.10.5. Observe that when a customer arrives, the server can be either idle I or busy B. Show that $\mathsf{P}(I) = 1 - \rho$.

1.10.6. If job n, say, finds the system empty, show that the expected time between the departure the $n$the job and job $n-1$ is given by $\mathsf{E}[D_n - D_{n-1}] = 1/\lambda + 1/\mu$.

1.10.7. Show that the density of $D_n - D_{n-1}$ is

$$f_{X+S}(t) = \frac{\lambda\mu}{\lambda - \mu}(e^{-\mu t} - e^{-\lambda t}).$$

1.10.8. Why is the probability that a job arrives at a busy station equal to ρ?

1.10.9. Show that the density of the inter-departure time if the server is busy is $f_X(t) = \mu e^{-\mu t}$.

1.10.10. Use conditioning to show that the density of the inter-departure time is $\lambda e^{-\lambda t}$

It can also be shown that the inter-departure times are independent.

1.10.11. Explain that we arrived at *Burke's law* which states that the departure process of the $M/M/1$ queue is a Poisson process with rate λ.

Relation between Inventory and Queueing Systems There exists an interesting relation between inventory and queueing systems, c.f. Figure 1.16. When a job arrives in the queueing system, the virtual workload $V(t)$ increases by the service time of the job. In the inventory system, the inventory $I(t)$ decreases by the demand size of the customer. Like this, the demand size of a customer at the inventory system converts into a production time (i.e., a service time) of a job at a queueing system. In the figure, the demand size D_1 of the first customer corresponds to a production time of duration $S_1 = D_1$, and so on. Thus, even though in the inventory system, customers do not have to wait, their demands spawn production times at a server who has to replenish the consumed items.

Assume now that the inventory process is controlled by an order-up-to policy: produce (refill the inventory) as long as the inventory level is below S and stop otherwise. Then the figure shows that the inventory level $I(t)$ is equal to $S - V(t)$.

In more general terms, in a queueing system or an inventory system, there is always 'something' or 'somebody' waiting. Items in a supermarket are produced ahead of the moment they are 'consumed', hence they 'wait' for customers. In a queueing system, customers are waiting while their product is 'produced' by the server. When there are no customers in a queueing system, the server waits until a new customers comes along. Thus, queueing and inventory theory are both focused on the distribution of waiting times, either by customers, servers, or items, hence both are related branches of (applied) probability theory.

1.10.12. Customers of fast-food restaurants prefer to be served from stock. For this reason such restaurants often use a 'produce-up-to' policy: When the on-hand inventory I is equal or lower than some threshold $S - 1$, the company

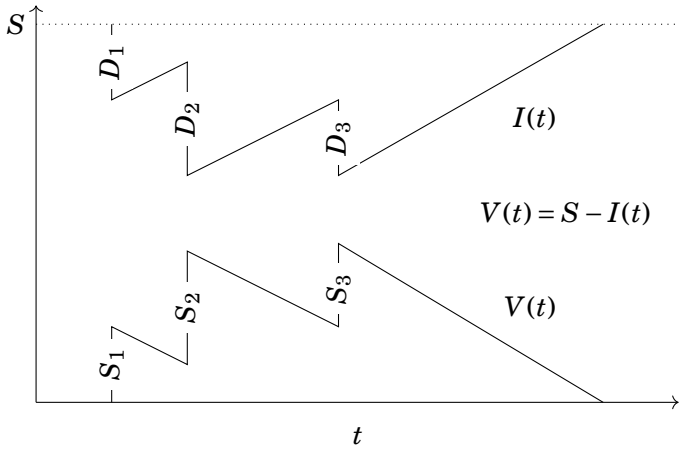


Figure 1.16: The relation between inventory and queueing systems. Here $I(t)$ models t evolution of the inventory level in an inventory system, while $V(t)$ shows the virtual wo load, and S is the order up to level. When a customer requires D_1 items, say, it take server of a time $S_1 = D_1$ to produce these items. Thus, demands at the inventory syst convert into production times in a queueing system. When the inventory is always repl ished to level S, then the shortage of the inventory level relative to S, i.e., $S - I(t)$, becon the workload $V(t)$ for the server in terms of amount of items or production time.

produces items until the inventory level equals S again. The level S is known as the order-up-to level, and $S-1$ as the reorder level.

Suppose that customers arrive as a Poisson process with rate λ and the production times of single items are i.i.d. and exponentially distributed with parameter μ. Assume also that customers who cannot be served from on-hand stock are backlogged, that is, they wait until their item has been produced. What are the average on-hand inventory level and the average number of customer in backlog?

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 such that the

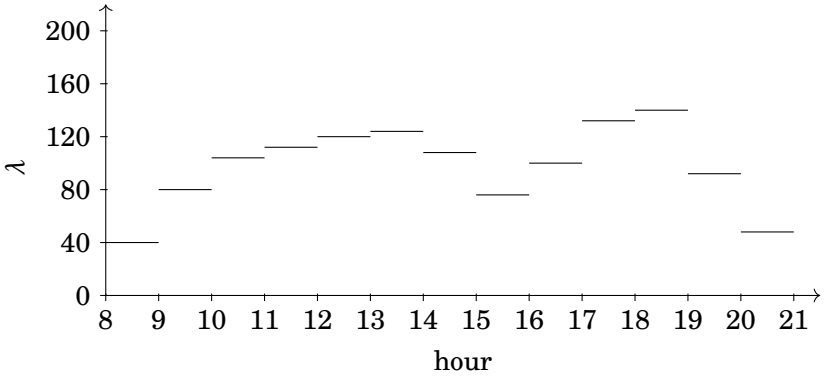


Figure 1.17: A demand profile of the arrival rate λ modeled as constant over each hour.

fraction of the time more than 10 people 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 small probability to go the supermarket on 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, c.f. Figure 1.17. Then we model the arrival process as Poisson with an arrival rate that is constant during a certain hour and is 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, and we just model the service time distribution as exponential with a mean of 1.5 minutes.

We also *model* the behavior of all the cashiers together (a multi-server queue) as a single fast server. Thus, we neglect any differences between a station with, for instance, 3 cashiers and a single server that works 3 times as fast as a normal cashier. As a 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:

$$c \geq \frac{\lambda \mathsf{E}[S]}{0.67} \approx \frac{3}{2} \cdot \lambda \cdot 1.5 \approx 2.5\lambda,$$

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 that $\lambda = 120$ customers per hour, hence $c = 2.5 \cdot 120/60 = 5$. 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 2.5.

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 not 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 $+$ indicate a working hour and $-$ a break of an hour. Then generate shift types for each of these plans with starting times 8am, 9am, and on 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 shift 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

$$\sum_i x_i \mathbb{1}[t \in s_i] \geq 2.5 \frac{\lambda_t}{60}$$

for all hours t the shop is open and λ_t is the demand for hour t.

Hints

h.1.10.1 There are various ways to get this result, one is with indicator functions, another with moment generating functions.

h.1.10.2 $\mathsf{P}(L \geq n) = \sum_{k \geq n} p(k)$.

h.1.10.6 After job $n-1$ left, job n has to arrive, so we need to wait first for this interarrival time. Then job n must be served. This adds up to $1/\lambda + 1/\mu$.

h.1.10.9 Use the memoryless property of the exponential distribution of the service times.

h.1.10.10 The final result is $f_D(t) = f_{X+S}(t)\mathsf{P}(I) + f_S \mathsf{P}(B) = (1-\rho) f_{X+S}(t) + \rho\mu e^{-\mu t}$. Now use the above exercises to simplify and see that $f_D(t) = \lambda e^{-\lambda t}$.

h.1.10.12 Use Figure 1.16 to realize that the inventory level $I(t)$ (here measured in the number of items on stock) at time t can be modeled as $I(t) = S + 1 - L(t)$, where L is the number of jobs in an $M/M/1$ queue. Note that the number of jobs in queue corresponds to the number of items to be produced. A customer of an item, i.e., a demand of the inventory system, turns into job at the queueing system, i.e., the demand becomes a job for the cook to replenish the item.

Solutions

s.1.10.1 With indicators, a bit long, but I spell out every step.

$$\begin{aligned}
\mathsf{E}[L] &= \sum_{n=0}^{\infty} n p(n) \\
&= \sum_{n=0}^{\infty} \sum_{i=1}^{n} \mathbb{1}[i \leq n] p(n) && n = \sum_{i=1}^{n} \mathbb{1}[i \leq n] \\
&= \sum_{n=0}^{\infty} \sum_{i=1}^{\infty} \mathbb{1}[i \leq n] p(n) && i > n \implies \mathbb{1}[i \leq n] = 0 \\
&= \sum_{i=1}^{\infty} \sum_{n=0}^{\infty} \mathbb{1}[i \leq n] p(n) && \text{Fubini}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{\infty} \sum_{n=i}^{\infty} p(n) && n < i \implies \mathbb{1}[i \le n] = 0 \\
&= \sum_{i=1}^{\infty} \sum_{n=i}^{\infty} (1-\rho)\rho^n && p(n) = (1-\rho)\rho^n \\
&= \sum_{i=1}^{\infty} \sum_{n=0}^{\infty} (1-\rho)\rho^{n+i} && n \to n+i \\
&= \sum_{i=1}^{\infty} (1-\rho)\rho^i \sum_{n=0}^{\infty} \rho^n && \rho^{n+i} = \rho^i \rho^n \\
&= \sum_{i=1}^{\infty} (1-\rho)\rho^i \frac{1}{1-\rho} \\
&= \sum_{i=1}^{\infty} \rho^i \\
&= \sum_{i=0}^{\infty} \rho^{i+1} && i \to i+1 \\
&= \rho \sum_{i=0}^{\infty} \rho^i \\
&= \frac{\rho}{1-\rho}.
\end{aligned}
$$

Note that, since the summands are positive, we can use Fubini's theorem to justify the interchange of the summations.

With moment generating functions it is a bit shorter.

$$
\begin{aligned}
M_L(s) = \mathsf{E}\left[e^{sL}\right] &= \sum_{n=0}^{\infty} e^{sn} p(n) = (1-\rho) \sum_n e^{sn} \rho^n \\
&= \frac{1-\rho}{1-e^s\rho},
\end{aligned}
$$

where we assume that s is such that $e^s\rho < 1$. Then,

$$
M_L'(s) = (1-\rho)\frac{1}{(1-e^s\rho)^2} e^s \rho.
$$

Hence, $\mathsf{E}[L] = M_L'(0) = \rho/(1-\rho)$.

s.1.10.2

$$
\begin{aligned}
\mathsf{P}(L \ge n) &= \sum_{k=n}^{\infty} p(k) = \sum_{k=n}^{\infty} p(0)\rho^k = (1-\rho) \sum_{k=n}^{\infty} \rho^k \\
&= (1-\rho)\rho^n \sum_{k=0}^{\infty} \rho^k = (1-\rho)\rho^n \frac{1}{1-\rho}.
\end{aligned}
$$

s.1.10.3 Jobs arrive at rate λ. For a stable queue, $\mu > \lambda$. Moreover, jobs can never leave faster than they arrive.

s.1.10.4 Because jobs do not leave at rate μ.

s.1.10.5 $\mathsf{P}(I) = p_0 = \pi_0$. Recall that π_0 is what the jobs see upon arrival and $p(0)$ is the time-average. By PASTA these are the same. Recall also that $p_0 = 1 - \rho$.

s.1.10.6 When job n finds an empty system, job $n-1$ left an empty system behind. Thus, we first have to wait for an inter-arrival time X_n. Then, since job n's service starts right away, it leaves when $D_n = D_{n-1} + X_n + S_n$. Thus, the expected duration is $\mathsf{E}[X_n + S_n] = 1/\lambda + 1/\mu$.

s.1.10.7 By the previous point, the density of $D_n - D_{n-1}$ is the same as the density of $X_n + S_n$. Therefore, by conditioning,

$$\begin{aligned} f_{X+S}(t) &= \mathsf{P}(X + S \in \mathrm{d}t) \\ &= \int \mathsf{P}(S + x \in \mathrm{d}t)\mathsf{P}(X \in \mathrm{d}x) \\ &= \int_0^t f_S(t-x) f_X(x)\,\mathrm{d}x \\ &= \int_0^t \mu e^{-\mu(t-x)} \lambda e^{-\lambda x}\,\mathrm{d}x \\ &= \lambda\mu e^{-\mu t} \int_0^t e^{x(\mu-\lambda)}\,\mathrm{d}x \\ &\frac{\lambda\mu}{\lambda-\mu}(e^{-\mu t} - e^{-\lambda t}). \end{aligned}$$

s.1.10.8 $\mathsf{P}(B) = \mathsf{P}(\text{busy}) = \rho$.

s.1.10.9 If the server is busy when a job arrives, the density of the time to the next departure epoch must be exponential. By the memoryless property, we may assume that the service restarts at an arrival epoch. Therefore, an arrival 'sees' $\mu e^{-\mu t}$ as the density of the departure time when the server is busy.

s.1.10.10

$$\begin{aligned} f_D(t) &= f_{X+S}(t)\mathsf{P}(I) + f_S\mathsf{P}(B) \\ &= (1-\rho) f_{X+S}(t) + \rho\mu e^{-\mu t} \end{aligned}$$

$$
\begin{aligned}
&= (1-\rho)\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t}-e^{-\lambda t}\right)+\rho\mu e^{-\mu t}\\
&= \left(1-\frac{\lambda}{\mu}\right)\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t}-e^{-\lambda t}\right)+\rho\mu e^{-\mu t}\\
&= \frac{\mu-\lambda}{\mu}\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t}-e^{-\lambda t}\right)+\frac{\lambda}{\mu}\mu e^{-\mu t}\\
&= -\lambda\left(e^{-\mu t}-e^{-\lambda t}\right)+\lambda e^{-\mu t}\\
&= \lambda e^{-\lambda t}.
\end{aligned}
$$

s.1.10.11 The above exercises show that inter-departures times have the same density, i.e., $\lambda e^{-\lambda t}$. The remark above states these times are independent. Thus, the inter-departures times form a set of i.i.d. exponentially distributed random variables with mean $1/\lambda$. Consequently, the departures times form a Poisson process with rate λ.

s.1.10.12 In the $M/M/1$ queue, $p(n)$ corresponds to the fraction of time there are n jobs in the system. In the inventory system, it means that the cook has n jobs to satisfy, hence the inventory is n jobs short relative to the order up to level S. Hence, the average on-hand inventory is S minus the average number of jobs at the cook. i.e., $\mathsf{E}[I] = \sum_{i=0}^{S}(S-i)p(i)$. The average number of customers in backlog is the fraction of time there are more than S replenishment jobs at the cook, i.e., $\mathsf{E}[B] = \sum_{i=S+1}^{\infty}(i-S)p(i)$.

1.11 M(n)/M(n)/1 Queue

Theory and Exercises

As it turns out, many more single-server queueing situations can be modeled and analyzed by making a judicious choice of $\lambda(n)$ and $\mu(n)$ in (1.45), to wit, $\lambda(n)p(n) = \mu(n+1)p(n+1)$. 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 interarrival times and service times need to be memoryless for the analysis below; the rates, however, may depend on the number of the jobs in the system. Specifically, consider the departure time D_k of the kth job, so that $A_{A(D_k)+1}$ is the arrival time of the next job. If $L(D_k) = n$, then we require that for every k

$$
\mathsf{P}\left(A_{A(D_k)+1} - D_k \le x\right) = 1 - e^{-f(n)\lambda x},
$$

for some function $f : \mathbb{N} \to [0,\infty)$. Next, since $D_{D(A_k)+1}$ is the time until the next departure after A_k, we assume for all k,

$$\mathsf{P}\left(D_{D(A_k)+1} - A_k \le x\right) = 1 - e^{-g(n)\mu x},$$

if $L(A_k) = n$ (not $L(A_k-) = n$) for some function $g : \mathbb{N} \to [0,\infty)$.

For the $M/M/1/K$, i.e., a system that cannot contain more than K jobs, take

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n < K, \\ 0, & \text{if } n \ge K, \end{cases}$$
$$\mu(n) = \mu.$$

Then,

$$p(n) = \frac{\rho^n}{G}, \quad 0 \le n \le K, \tag{1.51a}$$
$$G = \frac{1-\rho^{K+1}}{1-\rho}, \tag{1.51b}$$
$$P_{\text{loss}} = \mathsf{P}(L = K) = \frac{\rho^K}{G} = \frac{1-\rho}{1-\rho^{K+1}}\rho^K. \tag{1.51c}$$

1.11.1. ($M/M/1/K$ queue) Derive Eq. (1.51)

1.11.2. Show that as $K \to \infty$, the performance measures of the $M/M/1/K$ converge to those of the $M/M/1$ queue.

For the $M/M/c$ queue we can take

$$\lambda(n) = \lambda,$$
$$\mu(n) = \begin{cases} n\mu, & \text{if } n \le c, \\ c\mu, & \text{if } n \ge c. \end{cases}$$

This model is also known as the *Erlang C*-formula. Define the load as

$$\rho = \frac{\lambda}{c\mu}. \tag{1.52a}$$

Then,

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!}, \quad n = 0, \ldots, c, \tag{1.52b}$$

$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!}, \quad n = c, c+1, \ldots \tag{1.52c}$$

$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!}, \tag{1.52d}$$

$$\mathsf{E}\left[L_Q\right] = \sum_{n=c}^{\infty}(n-c)p(n) = \frac{(c\rho)^c}{c!G}\frac{\rho}{(1-\rho)^2}, \tag{1.52e}$$

$$\mathsf{E}[L_S] = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) = \frac{\lambda}{\mu}. \tag{1.52f}$$

1.11.3. Derive Eq.(1.52) for the $M/M/c$ queue.

1.11.4. Check that Eq.(1.52) for the $M/M/c$ queue reduces to the $M/M/1$ queue if $c = 1$.

From this we can easily get the $M/M/c/c$ queue; here jobs cannot be in queue, only in service, and the system has c servers. 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.

1.11.5. Find $\lambda(n)$ and $\mu(n)$ for the $M/M/c/c$ queue, and determine the performance measures.

1.11.6. Consider the $M/M/2/3$ queue with arrival rate λ and service rate μ (thus, at most 2 jobs can be in service and 1 in queue.) Derive first the level-crossing equations for this queueing system, then derive a simple and closed form expressions for the state probabilities in steady state.

1.11.7. (Multi-server queue with blocking) Consider the $M/M/c/c+K$ queue in which at most c jobs can be in service and K in queue. Try to derive the steady state probabilities $p(0)$, $p(1),\ldots$. You do not have to compute the normalization constant G.

From the $M/M/c$ queue (or the $M/M/c/c$ queue) we can also obtain the $M/M/\infty$, i.e., a queueing system with ample servers. By taking the limit $c \to \infty$, note first that in (1.52d),

$$\frac{(c\rho)^c}{(1-\rho)c!} = \frac{(\lambda/\mu)^c}{(1-\rho)c!} \to 0, \quad \text{as } c \to \infty.$$

Hence

$$G = \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} \to \sum_{n=0}^{\infty} \frac{(c\rho)^n}{n!} = e^{\lambda/\mu}.$$

Next, for $n \geq c$:

$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!} = \frac{\rho^n}{G}\frac{c^c}{c!} \to 0, \quad \text{as } c \to \infty,$$

since, from a standard limit $n^n/n! \to 0$ as $n \to \infty$. Therefore, for $n < c$:

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!} = \frac{1}{G}\frac{(\lambda/\mu)^n}{n!} \to e^{-\lambda/\mu}\frac{(\lambda/\mu)^n}{n!}, \quad \text{as } c \to \infty.$$

We see that the number of busy servers in the $M/M/\infty$ queue is Poisson distributed with parameter λ/μ, and $\mathsf{E}[L] = \mathsf{E}[L_S] = \lambda/\mu$. Observe that now λ/μ has no longer the interpretation of the fraction of time the server(s) are busy; it is the average number of busy servers.

We mention in passing—but do not prove it—that the same results also hold for the $M/G/\infty$ queue with $\lambda\,\mathsf{E}[S]$ rather than λ/μ.

1.11.8. Show that the $M/M/c$ queue converges to the $M/M/\infty$ queue as $c \to \infty$.

1.11.9. Show that the $M/M\infty$ queue is stable for any finite λ.

1.11.10. Why is $\mathsf{E}[L] = \rho$ for the $M/M/\infty$ queue?

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 x

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n = 0, \\ \lambda/2, & \text{if } n = 1, \\ \lambda/4, & \text{if } n = 2, \\ 0, & \text{if } n > 2, \end{cases}$$

and $\mu(n) = \mu$.

Observe that in the example with balking we made a subtle implicit assumption; in Section 1.12 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 $\mathsf{P}(L \leq n)$ is not necessarily in accordance with the perception of customers. To relate these two 'views', i.e., time-average versus observer-average, we need a new concept, *PASTA*, to be developed in in Section 1.12.

1.11.11. (Hall 5.1) Give two examples of systems that ordinarily have a finite buffer size. Give two examples of systems that have a finite calling population.

1.11.12. 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?

1.11.13. (Systems with finite calling population) Derive the steady state probabilities $p(n)$ for a single-server queue with a finite calling population with N members, i.e., jobs that are in service cannot arrive to the system. Check the answer you obtained for the cases $N = 1$ and $N = 2$. Interpret the results.

1.11.14. Derive the steady state probabilities $p(n)$ for a queue with a finite calling population with N members and N servers, i.e., the number of servers in the queueing system is equal the size of the calling population.

1.11.15. What would be the difference between a multi-server queue and a single-server queue with a fast server? We can use the formulas for the $M/M/1$ queue and the $M/M/c$ queue to obtain some basic understanding of the difference. To this end, suppose 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 $\mu = 3 \cdot 2 = 6$. When is it OK to approximate the $M/M/c$ queue by an $M/M/1$ queue with a fast server?

Hints

h.1.11.2 Use that $\sum_{i=0}^{n} x^i = (1-x^{n+1})/(1-x)$. BTW, is it necessary for this expression to be true that $|x| < 1$? What should you require for $|x|$ when you want to take the limit $n \to \infty$?

h.1.11.3 Use $\lambda(n)p(n) = \mu(n+1)p(n+1) = \min\{c, n+1\}\mu p(n+1)$.

h.1.11.4 Fill in $c = 1$. Realize that this is a check on the formulas.

h.1.11.6 Think about what would be the appropriate model choices for $\lambda(n)$ and $\mu(n)$ and use the level-crossing equations $\lambda(n)p(n) = \mu(n+1)p(n+1)$. For instance, realize that $\lambda(3) = 0$: the system cannot contain more than 3 jobs, hence a state with 4 jobs must be impossible. We can achieve that by setting $\lambda(3) = 0$. For the service rate, how many servers are busy when the system contains 2 or more jobs? What does this say about $\mu(k)$ for $k = 2$ or $k = 3$.

h.1.11.7 Use $\lambda(n)p(n) = \mu(n+1)p(n+1)$ and find suitable expressions for $\lambda(n)$ and $\mu(n+1)$.

h.1.11.8 Use that for any x, $x^n/n! \to 0$ as $n \to \infty$.

h.1.11.13 Use $\lambda(n)p(n) = \mu(n+1)p(n+1)$, and realize that for this case $\lambda(n) = (N-n)\lambda$ and $\mu(n) = \mu$.

h.1.11.15 Implement the formulas for $\mathsf{E}\left[L_Q(M/M/3)\right]$ for the $M/M/3$ queue in some computer program (R, excel, python, whatever) and compare this to $\mathsf{E}\left[L_Q(M/M/1)\right]$ for the fast $M/M/1$ case. Make plots of $\mathsf{E}\left[L_Q(M/M/3)\right]$ and $\mathsf{E}[L(M/M/3)]$ as functions of ρ.

Solutions

s.1.11.1 Note that

$$1 = \sum_{i=0}^{K} p(i) = p(0) \sum_{i=0}^{K} \rho^i = p(0) \frac{1-\rho^{K+1}}{1-\rho}.$$

Thus, for the normalization constant G we take

$$G = \frac{1}{p(0)} = \frac{1-\rho^{K+1}}{1-\rho},$$

and the result follows.

s.1.11.2 To take the limit $K \to \infty$—mind, not the limit $n \to \infty$—, write

$$G = \frac{1-\rho^{K+1}}{1-\rho} = \frac{1}{1-\rho} - \frac{\rho^{K+1}}{1-\rho}.$$

Since $\rho^{K+1} \to 0$ as $K \to \infty$ (recall, $\rho < 1$), we get

$$G \to \frac{1}{1-\rho},$$

as $K \to \infty$. Therefore $p(n) = \rho^n/G \to \rho^n(1-\rho)$, and the latter are the steady-state probabilities of the $M/M/1$ queue. Finally, if the steady state probabilities are the same, the performance measures (which are derived from $p(n)$) must be the same.

s.1.11.3 First we use the hint to establish a generic relation for $p(n+1)$:

$$\begin{aligned} p(n+1) &= \frac{\lambda}{\mu(n+1)} p(n) = \frac{\lambda}{\min\{c, n+1\}\mu} p(n) \\ &= \frac{1}{\min\{c, n+1\}} \frac{\lambda}{\mu} p(n) = \frac{1}{\min\{c, n+1\}} (c\rho) p(n) \\ &= \frac{1}{\min\{c, n+1\}\min\{c, n\}} (c\rho)^2 p(n-1) = \frac{1}{\Pi_{k=1}^{n+1} \min\{c, k\}} (c\rho)^{n+1} p(0). \end{aligned}$$

Thus, if $n < c$:

$$p(n) = \frac{1}{\Pi_{k=1}^{n} \min\{c, k\}} (c\rho)^n p(0) = \frac{(c\rho)^n}{n!} p(0),$$

since $\min\{c, k\} = k$ when $k < c$. If $n \geq c$:

$$p(n) = \frac{1}{\Pi_{k=1}^{n} \min\{c, k\}} (c\rho)^n p(0)$$

$$
\begin{aligned}
&= \frac{1}{\Pi_{k=1}^{c-1} k \cdot \Pi_{k=c}^{n} c} (c\rho)^n p(0) \\
&= \frac{1}{(c-1)! c^{n-c+1}} c^n \rho^n p(0) \\
&= \frac{1}{c! c^{n-c}} c^n \rho^n p(0) = \\
&= \frac{c^c}{c!} \rho^n p(0).
\end{aligned}
$$

For the normalization constant G set $p(0) = 1$ for the time being.

$$
\begin{aligned}
G = \sum_{n=0}^{\infty} p(n) &= \sum_{n=0}^{c-1} p(n) + \sum_{n=c}^{\infty} p(n) \\
&= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \sum_{n=c}^{\infty} \frac{c^c}{c!} \rho^n \\
&= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \sum_{n=c}^{\infty} \frac{(c\rho)^c}{c!} \rho^{n-c} \\
&= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!} \sum_{n=0}^{\infty} \rho^n \\
&= \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}.
\end{aligned}
$$

Next,

$$
\begin{aligned}
\mathsf{E}\left[L_Q\right] &= \sum_{n=c}^{\infty} (n-c) p(n) \\
&= \sum_{n=c}^{\infty} (n-c) \frac{c^c}{c!} \rho^n p(0) \\
&= \frac{c^c \rho^c}{Gc!} \sum_{n=c}^{\infty} (n-c) \rho^{n-c} \\
&= \frac{c^c \rho^c}{Gc!} \sum_{n=0}^{\infty} n \rho^n = \frac{c^c \rho^c}{Gc!} \frac{\rho}{(1-\rho)^2},
\end{aligned}
$$

where, with our common trick (if we don't want to use generating functions),

$$
\begin{aligned}
\sum_{n=0}^{\infty} n\rho^n &= \sum_{n=0}^{\infty} \sum_{i=1}^{\infty} \mathbb{1}[i \le n] \rho^n = \sum_{i=1}^{\infty} \sum_{n=0}^{\infty} \mathbb{1}[i \le n] \rho^n \\
&= \sum_{i=1}^{\infty} \sum_{n=i}^{\infty} \rho^n = \sum_{i=1}^{\infty} \rho^i \sum_{n=0}^{\infty} \rho^n
\end{aligned}
$$

$$= \frac{1}{1-\rho}\sum_{i=1}^{\infty}\rho^i = \frac{\rho}{1-\rho}\sum_{i=0}^{\infty}\rho^i = \frac{\rho}{(1-\rho)^2}.$$

Observe again that using indicators and Fubini's theorem (interchanging summations and integrals) makes the above computation painless. Realize, by the way, that

$$\sum_{n=0}^{\infty} np(n) = \sum_{n=1}^{\infty} np(n).$$

We next show that the expected number of jobs in service is given by

$$\mathsf{E}[L_S] = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n).$$

This expression is not the easiest to start with. With a slight modification the entire derivation becomes easier. I also pre-multiply by the normalization constant G to get rid of it on the right hand side.

$$\begin{aligned} G\,\mathsf{E}[L_S] &= G\sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) \\ &= \sum_{n=1}^{c} n\frac{(c\rho)^n}{n!} + \sum_{n=c+1}^{\infty} c\frac{c^c\rho^n}{c!} = \sum_{n=1}^{c}\frac{(c\rho)^n}{(n-1)!} + \frac{c^{c+1}}{c!}\sum_{n=c+1}^{\infty}\rho^n \\ &= \sum_{n=0}^{c-1}\frac{(c\rho)^{n+1}}{n!} + \frac{(c\rho)^{c+1}}{c!}\sum_{n=0}^{\infty}\rho^n = c\rho\left(\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}\right). \end{aligned}$$

Observe that the right hand side is precisely equal to ρcG, and hence,

$$\mathsf{E}[L_S] = c\rho = \frac{\lambda}{\mu}.$$

s.1.11.4 Take $c = 1$

$$p(0) = \frac{1}{G}\frac{(c\rho)^0}{0!} = \frac{1}{G}, \quad n = 0,\ldots,1-1 \tag{1.53}$$

$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!} = \frac{1}{G}\frac{1^1\rho^n}{1!} = \frac{\rho^n}{G}, \quad n = 1,1+1,\ldots \tag{1.54}$$

$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} = \sum_{n=0}^{0}\frac{\rho^0}{0!} + \frac{\rho}{(1-\rho)} = 1 + \frac{\rho}{1-\rho} = \frac{1}{1-\rho} \tag{1.55}$$

$$\mathsf{E}[L_Q] = \frac{(c\rho)^c}{c!G}\frac{\rho}{(1-\rho)^2} == \frac{\rho}{1/(1-\rho)}\frac{\rho}{(1-\rho)^2} = \frac{\rho^2}{1-\rho} \tag{1.56}$$

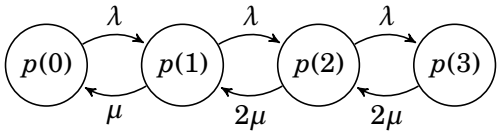
$$\mathsf{E}[L_S] = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) = p(1) + 1\sum_{n=2}^{\infty} p(n) = 1 - p(0) = \rho. \tag{1.57}$$

Everything is in accordance to the formulas we derived earlier for the $M/M/1$ queue.

s.1.11.5 $\lambda(n) = \lambda$ if $n < c$, and $\lambda(n) = 0$ for $n \geq c$. Then, $\mu(n) = n\mu$ for $n \leq c$. (And n can never be larger than c, since $\lambda(n) = 0$ for $n \geq c$.) Define $\rho = \lambda/c\mu$. $p(n) = G^{-1}(c\rho)^n/n!$, $G = \sum_{n=0}^{c}(c\rho)^n/n!$. $\mathsf{E}[L_Q] = 0$.

$$\mathsf{E}[L_S] = \sum_{n=0}^{c} np(n) = G^{-1}\sum_{n=0}^{c} n(\lambda/\mu)^n/n!$$
$$G^{-1}\lambda/\mu\sum_{n=0}^{c-1}(\lambda/\mu)^{n-1}/(n-1)! =$$
$$\frac{\lambda}{\mu}\frac{G-(\lambda/\mu)^c/c!}{G}.$$

s.1.11.6 Use the figure below. Make sure you understand why $\mu(2) = 2\mu$ and so on.



From this figure it follows right away that:

$$\lambda p(0) = \mu p(1)$$
$$\lambda p(1) = 2\mu p(2)$$
$$\lambda p(2) = 2\mu p(3)$$

Then, from the above, with $\rho = \lambda/\mu$:

$$p(1) = \rho p(0),$$
$$p(2) = (\rho/2)p(1) = (\rho^2/2)p(0),$$
$$p(3) = (\rho/2)p(2) = (\rho^3/4)p(0).$$

Now we normalize to find $p(0)$. Thus, we want that:

$$1 = p(0) + p(1) + p(2) + p(3) = p(0)\left(1 + \rho + \frac{\rho^2}{2} + \frac{\rho^3}{4}\right),$$

hence,

$$p(0) = (1 + \rho + \rho^2/2 + \rho^3/4)^{-1}.$$

s.1.11.7 $\lambda(n) \equiv \lambda$ for all $n < c+K$. When $n = c+K$, $\lambda(n) = 0$, since then the system is full, and all arriving jobs will be dropped; in other words, there will still be jobs arriving to the system when $L = c+K$, but these jobs will be rejected, hence cannot generate a transition from state $c+K$ to $c+K+1$. When $n < c$, $\mu(n) = n\mu$ since only n servers are active/occupied when the system contains n jobs. When $n \geq c$, $\mu(n) = c\mu$. Thus, using $\rho = \lambda/(c\mu)$, for $n < c$,

$$p(n) = \frac{\lambda}{n\mu} p(n-1) = \frac{(\lambda/\mu)^n}{n!} p(0) = \frac{(c\rho)^n}{n!} p(0).$$

For $c \leq n \leq c+K$ and using the above to get $p(c-1)$:

$$\begin{aligned} p(n) &= \frac{\lambda}{c\mu} p(n-1) = \rho p(n-1) = \rho^2 p(n-2) = \dots \\ &= \rho^{n-c+1} p(c-1) = \rho^{n-c+1} \frac{(c\rho)^{c-1}}{(c-1)!} p(0) \\ &= \rho^n \frac{(c)^{c-1}}{(c-1)!} p(0) = \rho^n \frac{(c)^{c-1} c}{(c-1)! c} p(0) = \frac{c^c \rho^n}{c!} p(0). \end{aligned}$$

The normalization is trivial, numerically at least.

s.1.11.8 The second term in (1.52d) is $(c\rho)^c/c! = (\lambda/\mu)^c/c!$. It is well known that $x^c/c! \to 0$ as $c \to \infty$.

s.1.11.9 No matter how many jobs are in service, there is always another free server available when a new job arrives. Thus, jobs never have to wait in queue, and only spend time in service. Since $\mathsf{E}[S] < \infty$ by assumption, jobs spend a finite time (with probability one) at a server.

s.1.11.10 Write $\rho = \lambda/\mu$. Then, from the formulas for the $M/M/\infty$ queue, it follows that $p(n) = e^{-\rho}\rho^n/n!$. Interestingly, we see that this is equal to $\mathsf{P}(N = n)$ where N is a Poisson r.v. with parameter ρ. Thus, the number in the system L is Poisson distributed with parameter ρ, thus $\mathsf{E}[L] = \rho$.

Another way to see that $\mathsf{E}[L] = \rho$ is by noting that in the $M/M/\infty$ queue jobs do not interact with each other in the queue. When they arrive, there is always a free server available. Since work arrives at rate ρ, and all jobs are in service simultaneously, the average number of busy servers must also be ρ.

s.1.11.11 Finite buffer size. Formally the number of customers that fit into a shop is necessarily finite. This answer, however, is not intended. Typically, the number

of customers in a restaurant is limited. Example 2: Sometimes call centers reject callers when the system is too busy.

A finite calling population occurs for instance at a factory with a number of machines. When a machine breaks down, it becomes a (repair) job at the repair department. Thus, a break down forms an arrival at the repair shop. The mechanics at the repair department form a set of parallel servers. Typically, the number of machines quite small, 10 or so, and when a machine is 'down', i.e., broken, it cannot break again. Hence, when 2, say, machines are in repair, the number of 'customers' that can arrive to the queueing system is only 8.

s.1.11.12 In a queueing system with balking, customers may decide to balk at a level b. Thus, whether only b customers are admitted to the system (i.e., blocked), or balk at level b, the effect is the same: the number of people in the system remains at or below b. However, a fraction of customer may already balk at lower levels, like in the example above, so that the arrival stream is 'thinned' due to balking customers. In that respect, a queueing system with balking behaves differently.

s.1.11.13 Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = \mu$, and solve Eq. (1.45) and (1.47). Thus:

$$\begin{aligned} p(n+1) &= \frac{(N-n)\lambda}{\mu} p(n) = \rho(N-n)p(n) \\ &= \rho^2(N-n)(N-(n-1))p(n-1) \\ &= \rho^3(N-n)(N-(n-1))(N-(n-2))p(n-2) \\ &= \rho^{n+1}(N-n)(N-(n-1))\cdots(N-(0))p(0) \\ &= \rho^{n+1}\frac{N!}{(N-(n+1))!}p(0). \end{aligned}$$

Next, we need to normalize this. Observe that $p(N+1) = P(N+2) = \ldots = 0$ since there are just N customers, so that the system can never contain more than N customers. Thus, we want $p(0)$ to be such that

$$1 = \sum_{n=0}^{N} p(n) = p(0) \sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!}$$

We see from this that $p(0)$ times some constant must be 1. Hence, dividing by this constant, we get

$$p(0) = \left(\sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!} \right)^{-1}.$$

I asked WolframAlpha to simplify this, but the answer I got was not particularly revealing.

s.1.11.14 Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = n\mu$. Then

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)} p(n) = \frac{(N-n)\lambda}{(n+1)\mu} p(n) = \frac{(N-n)(N-(n-1))}{(n+1)n} \frac{\lambda^2}{\mu^2} p(n-1)$$
$$= \frac{N!}{(N-(n+1))!} \frac{1}{(n+1)!} \rho^{n+1} p(0) = \binom{N}{n+1} \rho^{n+1} p(0).$$

Hence, after normalization, i.e., requiring that $p(0)$ is such that $\sum_{n=0}^{N} p(n) = 1$, so that $p(0) = \left(\sum_{k=0}^{N} \rho^k \binom{N}{k}\right)^{-1}$, the final result becomes

$$p(n) = \frac{\rho^n \binom{N}{n}}{\sum_{k=0}^{N} \rho^k \binom{N}{k}}.$$

s.1.11.15 I am going to implement the formulas of Eq. (1.52) in python. First the results for the $M/M/3$ queue.

```
>>> from math import exp, factorial

>>> labda = 5
>>> mu = 2
>>> c = 3

>>> rho = labda / mu / c
>>> rho
0.8333333333333334

>>> G = sum((c * rho)**n / factorial(n) for n in range(c))
>>> G += (c * rho)**c / ((1 - rho) * factorial(c))
>>> G
22.250000000000004

>>> ELQ = (c * rho)**c / (factorial(c) * G) * rho / (1 - rho)**2
>>> ELQ
3.511235955056181
>>> ELS = rho * c
>>> ELS
2.5
>>> EL = ELQ + ELS
>>> EL
6.011235955056181
```

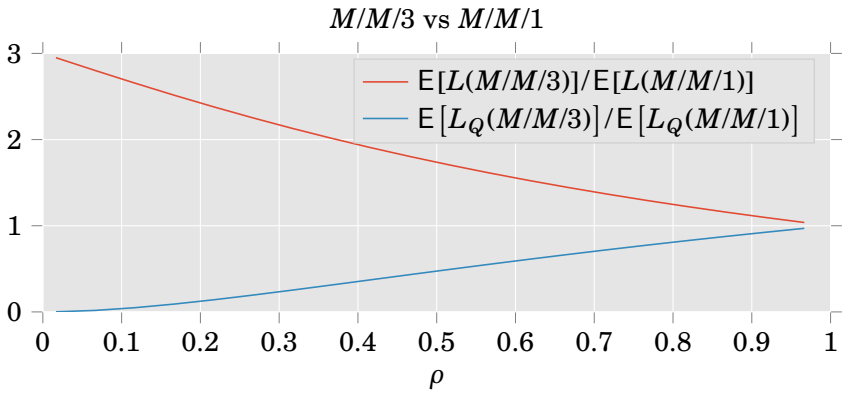
Now for the $M/M/1$ queue:

```
>>> labda = 5
>>> c = 3
>>> mu = 2*c

>>> rho = labda / mu
>>> rho
0.8333333333333334

>>> ELQ = rho**2/(1-rho)
>>> ELQ
4.166666666666668
>>> ELS = rho
>>> ELS
0.8333333333333334
>>> EL = ELS + ELQ
>>> EL
5.000000000000001

>>> rho/(1-rho) # this must also be EL, just a check
5.000000000000002
```

Note the last check. As a rule, you should always compare your results with known results. BTW, that is one of the reasons I prefer to code the formulas instead of using a calculator. Testing with code is relatively easy, whereas with a calculator it is impossible (You simply can't check what you typed at the calculator.)

So, returning to the results, as expected, the number of jobs in queue is smaller for the $M/M/3$ queue, but the number in service is higher.

To put things in a larger perspective, see the figure below where we plot the ratio of the queue lengths and the system length as functions of ρ. We see, in case of high load, that $\mathsf{E}\left[L_Q\right]$ and $\mathsf{E}[L]$ are nearly the same for both systems. This is as expected: when the load is high, most jobs should be in the queue. Therefore, $\mathsf{E}\left[L_Q\right]/\mathsf{E}[L] \to 1$ as $\rho \to 1$. When ρ is small, the difference is quite large. This is also reasonable, because the service time in the fast $M/M/1$ is 3 times as small as the service time in the $M/M/3$ queue. Hence, as ρ is small, the time in the system is dominated by service time, as there is hardly any queueing time, if at all. Thus, there must be more jobs in the system on average in the $M/M/3$ queue than in the fast $M/M/1$ queue.



The code can be found on `github` in the `progs` directory.

1.12 Poisson Arrivals See Time Averages

Theory and Exercises

Suppose the following limit exists:

$$\pi(n) = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}[L(A_k-) = n], \tag{1.58}$$

where $\pi(n)$ is the long-run fraction of jobs that observe n customers in the system at the moment of arrival. It is natural to ask whether $\pi(n)$ and $p(n)$, as defined by (1.41), 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 $\pi(n) = p(n)$.

We can make some progress by rewriting $\pi(n)$ in the following way. Since $A(t) \to \infty$ as $t \to \infty$, it is reasonable that[4]

$$\begin{aligned}
\pi(n) &= \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{A(t)} \mathbb{1}[L(A_k-) = n] \\
&= \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{\infty} \mathbb{1}[A_k \le t, L(A_k-) = n] \\
&= \lim_{t\to\infty} \frac{A(n,t)}{A(t)},
\end{aligned} \tag{1.59}$$

[4] See below for the proof.

where we use (1.40a) in the last row. But,

$$\frac{A(n,t)}{t} = \frac{A(t)}{t}\frac{A(n,t)}{A(t)} \to \lambda\pi(n), \quad \text{as } t \to \infty, \tag{1.60}$$

where we use Eq. (1.28), i.e., $A(t)/t \to \lambda$. Next, by Eq. (1.43),

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

Thus

$$\lambda\pi(n) = \lambda(n)p(n), \tag{1.61}$$

from which follows our final result:

$$\lambda(n) = \lambda \iff \pi(n) = p(n),$$

1.12.1. Show that for the case of Exercises 1.9.6 and 1.9.7. $\pi(0) = 1$, $\pi(n) = 0$, for $n > 0$.

1.12.2. Check that (1.61) holds for the system of Exercise 1.12.1

So, why is this useful? Well, in words, it means that if the arrival rate does not depend on the state of the system, i.e., $\lambda = \lambda(n)$, the sample average $\pi(n)$ is equal to the time-average $p(n)$, i.e., $\pi(n) = p(n)$. But, when $\pi(n) = p(n)$, the customer perception at arrival moments, i.e., $\pi(n)$, is the same as the server perception, i.e., $p(n)$.

At the exercises above show, this property is not satisfied in general. However, when the arrival process is Poisson we have that $\lambda(n) = \lambda$. This fact is typically called *PASTA: Poisson Arrivals See Time Averages*. Thus, when customers arrive in accordance to a Poisson process (hence the inter-arrival times are exponentially distributed), it must be that $\pi(n) = p(n)$, and for the $M/M/1$ queue,

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

With the above reasoning, we can also establish a relation between $\pi(n)$ and the statistics of the system as obtained by the departures. For this we turn again to Eq. (1.42), i.e., $|A(n,t) - D(n,t)| \le 1$. To obtain Eq. (1.44) we divided both sides of this

equation by the time the system spends in a certain state. We can also use another form:

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

Taking limits at the left and right, we see again that the left hand becomes $\lambda\pi(n)$. For the right hand side, we use Eq. (1.30) and define, analogous to (1.59),

$$\delta(n) = \lim_{t\to\infty} \frac{D(n,t)}{D(t)}. \tag{1.62}$$

Thus, $\delta(n)$ is the long-run fraction of jobs that leave n jobs *behind*. Clearly, then, if the limits exist, the right hand side tends to $\delta\delta(n)$ as $t \to \infty$. Hence, for (queueing) systems in which customers arrive and leave as single units, we have

$$\lambda\pi(n) = \delta\delta(n). \tag{1.63}$$

Moreover, if the system is rate-stable, i.e., the output rate δ is equal to the input rate λ, we obtain

$$\lambda = \delta \iff \pi(n) = \delta(n). \tag{1.64}$$

This means that the system as seen by arrivals, i.e., $\pi(n)$, is the same as what jobs leave behind, i.e., $\delta(n)$.

1.12.3. When $\lambda \neq \delta$, is $\pi(n) \geq \delta(n)$?

1.12.4. Show that

$$\lambda\pi(n) = \lambda(n)p(n) = \mu(n+1)p(n+1) = \delta\delta(n).$$

What is the important condition for this to be true?

1.12.5. Why is $\mu(n) = \mu$ for the $M/M/1$ queue?

1.12.6. Use the PASTA property and the ideas of Section 1.9 to derive for the $M/M/1$ queue that $(\lambda+\mu)\pi(n) = \lambda\pi(n-1) + \mu\pi(n+1)$

There is a subtle problem in the transition from (1.58) to (1.59) and the derivation of (1.60): $\pi(n)$ 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. The resolution lies in the *renewal reward theorem*. This theorem is very useful in its own right, 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 1.18 provides graphical motivation about why this theorem is true; El-Taha and Stidham Jr. [1998] gives a (simple) proof.

Theorem 1.12.1 (Renewal Reward Theorem, $Y = \lambda X$)**.** Suppose the counting process $\{N(t), t \geq 0\}$ is such that $N(t)/t \to \lambda$ as $t \to \infty$, where $0 < \lambda < \infty$. Let $\{Y(t), t \geq 0\}$ be a non-decreasing right-continuous (deterministic) process. Define $X_k = Y(T_k) - Y(T_{k-1})$, $k \geq 1$, where T_k are the epochs at which N increases, i.e., $N(t) = \{k : T_k \leq t\}$. Then $Y(t)/t$ has a limit iff $n^{-1}\sum_k^n X_k$ has a limit in which case $Y = \lambda X$, i.e.,

$$\lim_t \frac{Y(t)}{t} = Y \iff \lim_n \frac{1}{n}\sum_k^n X_k = X \implies Y = \lambda X.$$

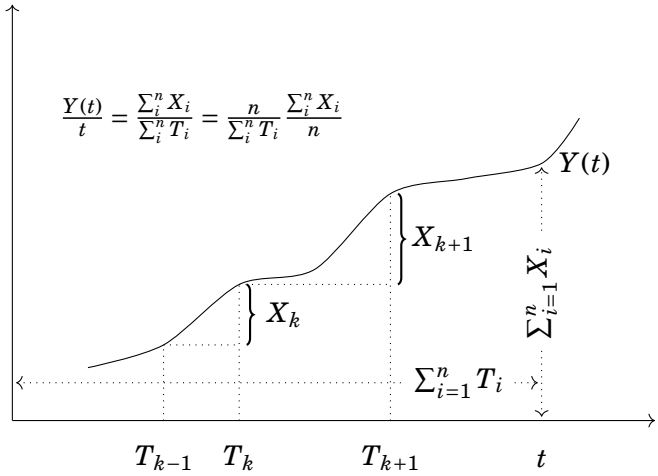


Figure 1.18: A graphical 'proof' of $Y = \lambda X$. Here $Y(t)/t \to Y$, $n/\sum_i^n T_i \to \lambda$ and $n^{-1}\sum_i^n X_i$ X. (Observe that in the figure X_k does not represent an interarrival time; instead corresponds to the increment of (the graph of) $Y(t)$ between two consecutive epochs T_k and T_k at which $Y(t)$ is observed.)

1.12.7. Use the renewal reward theorem to show that (1.59) is valid.

Hints

h.1.12.3 Use that $\lambda \geq \delta$ always, and $\lambda = \delta$ when the system is rate-stable.

h.1.12.4 Check all definitions of $Y(n,t)/t$ and so on.

h.1.12.5 Think about the construction of the $M/M/1$ queue as a random walk, see Section 1.6.

h.1.12.6 Consider some state n and count all transitions that 'go in and out of' this state. Specifically, $A(n,t)+D(n-1,t)$ counts all transitions out of state n: $A(n,t)$ counts the number of arrivals that see n in the system upon arrival, hence immediately after such arrivals the system contains $n+1$ jobs; likewise, $D(n-1,t)$ counts all jobs that leave $n-1$ jobs behind, hence immediately before such jobs depart the system contains n jobs. In a similar way, $A(n-1,t)+D(n,t)$ counts all transitions into state n (Recall once again, $D(n,t)$ counts the jobs that leave n behind. Hence, when such departures occur, state n is entered). Now use that 'what goes in must go out'.

h.1.12.7 Check that the conditions of the renewal reward theorem are satisfied in the above proof (1.60). Then define

$$Y(t) := A(n,t) = \sum_{k=1}^{A(t)} \mathbb{1}[L(A_k-) = n]$$
$$X_k := Y(A_k) - Y(A_{k-1}) = A(n,A_k) - A(n,A_{k-1}) = \mathbb{1}[L(A_k-) = n].$$

Solutions

s.1.12.1 All arrivals see an empty system. Hence $A(0,t)/A(t) \approx (t/2)/(t/2) = 1$, and $A(n,t) = 0$ for $n > 0$. Thus, $\pi(0) = \lim_t A(0,t)/A(t) = 1$ and $\pi(n) = 0$ for $n > 0$. Recall from the other exercises that $p(0) = 1/2$. Hence, time average statistics are not the same as statistics at arrival moments.

s.1.12.2 From the relevant previous exercises, $\lambda = \lim_t A(t)/t = 1/2$. $\lambda(0) = 1$, $p(0) = 1/2$, and $\pi(0) = 1$. Hence,

$$\lambda\pi(0) = \lambda(0)p(0) \implies \frac{1}{2} \times 1 = 1 \times \frac{1}{2}.$$

For $n > 0$ its easy, everything is 0.

s.1.12.3 We have shown for one-step transition processes that $\lambda\pi(n) = \delta\delta(n)$. Thus, $\pi(n) = \delta(n)\delta/\lambda$. Since $\lambda \geq \delta$, we have that $\pi(n) \leq \delta(n)$.

s.1.12.4 The important condition is that transitions occur as single steps. In other words, the relation is true for processes with *one-step* transitions, i.e, when $|A(n,t) - D(n,t)| \leq 1$. In that case,

$$\begin{aligned}
\frac{A(n,t)}{t} &= \frac{A(n,t)}{A(t)}\frac{A(t)}{t} \to \pi(n)\lambda \\
\frac{A(n,t)}{t} &= \frac{A(n,t)}{Y(n,t)}\frac{Y(n,t)}{t} \to \lambda(n)p(n) \\
\frac{D(n,t)}{t} &= \frac{D(n,t)}{Y(n+1,t)}\frac{Y(n+1,t)}{t} \to \mu(n+1)p(n+1) \\
\frac{D(n,t)}{t} &= \frac{D(n,t)}{D(t)}\frac{D(t)}{t} \to \delta(n)\delta
\end{aligned}$$

s.1.12.5 The $M/M/1$-queue can be constructed as a reflection of the random walk $Z(t) = Z(0) + N_\lambda(t) - N_\mu(t)$. Clearly, downcrossings can only occur when N_μ fires. The rate at which the transitions of N_μ occur is constant, and, in particular, independent of the history of Z.

More specifically, for the interested, define $\sigma\{X(t) : t \in I\}$ as the σ-algebra generated by the stochastic procsses $\{X(t), t \in I\}$ on the index set I. Then, by construction of $\{N_\lambda(t)\}$ and $\{Z(t)\}$, we have that $\sigma\{Z(s) : s \in [0,t]\}$ and $\sigma\{N_\lambda(u) : u > t\}$ are independent.

s.1.12.6 By the hint, the difference between the 'in' transitions and the 'out' transitions is at most 1 for all t. Thus, we can write

$$A(n,t) + D(n-1,t) \approx A(n-1,t) + D(n,t)$$

From this,

$$\begin{aligned}
&A(n,t) + D(n-1,t) \approx A(n-1,t) + D(n,t) \iff \\
&\frac{A(n,t) + D(n-1,t)}{t} \approx \frac{A(n-1,t) + D(n,t)}{t} \iff \\
&\frac{A(n,t)}{t} + \frac{D(n-1,t)}{t} \approx \frac{A(n-1,t)}{t} + \frac{D(n,t)}{t}.
\end{aligned}$$

Using the ideas of Section 1.9 this becomes for $t \to \infty$,

$$(\lambda(n) + \mu(n))p(n) = \lambda(n-1)p(n-1) + \mu(n+1)p(n+1).$$

Since we are concerned here with the $M/M/1$ queue we have that $\lambda(n) = \lambda$ and $\mu(n) = \mu$, and using PASTA we have that $p(n) = \pi(n)$. We are done.

s.1.12.7 First we check the conditions. The counting process here is $\{A(t)\}$ and the epochs at which $A(t)$ increases are $\{A_k\}$. By assumption, $A_k \to \infty$, hence $A(t) \to \infty$ as $t \to \infty$. Moreover, by assumption $A(t)/t \to \lambda$. Also $A(n,t)$ is evidently non-decreasing and $A(n,t) \to \infty$ as $t \to \infty$.

From the definitions in the hint,

$$X = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} X_k = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}[L(A_k-) = n] = \pi(n).$$

Since $Y = \lim_t Y(t)/t = \lim_t A(n,t)/t$ it follows from the renewal reward theorem that

$$Y = \lambda X \implies \lim_t \frac{A(n,t)}{t} = \lambda X = \lambda\pi(n).$$

Thus, Eq. (1.60) follows from the renewal reward theorem.

1.13 Little's Law

Theory and Exercises

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*:

$$\mathsf{E}[L] = \lambda\,\mathsf{E}[W]. \tag{1.65}$$

Here we provide a sketch of its proof; El-Taha and Stidham Jr. [1998] provide the details. 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 with defining a few intuitively useful concepts. Clearly, from (1.37),

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

is the time-average of the number of jobs in the system during $[0,t]$. Observe once again from the second equation that $\int_0^t L(s)\,\mathrm{d}s$ is the area enclosed between the graphs of $A(s)$ and $D(s)$.

The waiting time of the kth job is the time between the moment the job arrives and departs, that is

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

We can actually relate W_k to $L(t)$, see Figure 1.7. Consider a departure time T at which the system is empty. Observe that $A(T) = D(T)$, as at time T all jobs that arrived up to T also have left. As for all jobs $k \le A(T)$ we have that $D_k \le T$, we can replace the integration bounds in the above expression for W_k by

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

Moreover, if $s \le T$,

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

1.13.1. Show that

$$\int_0^T L(s) \, \mathrm{d}s = \sum_{k=1}^{A(T)} W_k.$$

1.13.2. Use the result of the previous exercise to show (1.65).

1.13.3. Which assumptions have we used to prove Little's law?

1.13.4. Observe that the area between the graphs of $A(s)$ and $D(s)$ must be equal to the total waiting time spent by all jobs in the system until T. Use this to provide a graphical interpretation of the proof of Little's law.

1.13.5. Use the dimensions of the components of Little's law to check that $\mathsf{E}[W] \neq \lambda \mathsf{E}[L]$.

1.13.6. As a useful first application, consider the server of a $G/G/1$ queue as a system by itself. Show that the average number of jobs in service $\mathsf{E}[L_S]$ is equal to the utilization ρ.

1.13.7. 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. What is the average time customers spend in the system?

1.13.8. For the single server queueing system of the previous exercise, 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? Why is this answer different from the answer of the previous exercises?

Here is a summary of the most useful results and concepts of this and the previous sections

- Arrival rate, departure rate, rate stability: $\lambda = \delta$
- PASTA: $\lambda\pi(n) = \lambda(n)p(n)$,
- Recursions $\lambda(n)p(n) = \mu(n+1)p(n+1)$,
- Renewal reward: $Y = \lambda X$
- Little's law: $\mathsf{E}[L] = \lambda\,\mathsf{E}[W]$.

We will use these results time and again in the sequel.

Hints

h.1.13.1 Substitute the definition of $L(s)$ in the left hand side, then reverse the integral and summation.

h.1.13.2 Divide both sides by T. At the right hand side use that $1/T = A(T)/T \cdot 1/A(T)$. Take limits.

h.1.13.4 Make a drawing of $A(t)$ and $D(t)$ until time T, i.e., the first time the system is empty. Observe that $A(t)-D(t)$ is the number of jobs in the system. Take some level k, and compute $A_k = A^{-1}(k)$ and $D_k = D^{-1}(k)$. Observe that $D_k - A_k = D^{-1}(k) - A^{-1}(k)$ is the waiting time of job k.

h.1.13.5 Checking the dimensions in the formula prevents painful mistakes.

h.1.13.7 Start with checking the units when applying Little's law.

Solutions

s.1.13.1

$$\begin{aligned}\int_0^T L(s)\,\mathrm{d}s &= \int_0^T \sum_{k=1}^{A(T)} 1\{A_k \le s < D_k\}\,\mathrm{d}s \\ &= \sum_{k=1}^{A(T)} \int_0^T 1\{A_k \le s < D_k\}\,\mathrm{d}s = \sum_{k=1}^{A(T)} W_k.\end{aligned}$$

s.1.13.2 From the previous exercise:

$$\frac{1}{T}\int_0^T L(s)\,\mathrm{d}s = \frac{A(T)}{T}\frac{1}{A(T)}\sum_{k=1}^{A(T)} W_k.$$

Assuming there are an infinite number of times $0 \le T_i < T_{i+1} < \cdots$, $T_i \to \infty$, at which $A(T_i) = D(T_i)$ and the following limits exist

$$\frac{1}{T}\int_0^T L(s)\,\mathrm{d}s \to \mathsf{E}[L], \qquad \frac{A(T_i)}{T_i} \to \lambda, \qquad \frac{1}{A(T_i)}\sum_{k=1}^{A(T_i)} W_k \to \mathsf{E}[W],$$

we obtain Little's law.

s.1.13.3

- $A(t)/t \to \lambda$ as $t \to \infty$, i.e., $A(t)/t$ has a limit as t converges to ∞.
- There exists a sequence of points $T_k, k = 0,1,2,\ldots$ in time such that the server is idle.
- Either of the limits $\sum_k^n W_k/n = \sum_k^n S_k/n$ or $t^{-1}\int_0^t L(s)\,\mathrm{d}s$ exists, in which case the other exists.

s.1.13.4 The area enclosed between the graphs of $A(t)$ and $D(t)$ until T can be 'chopped up' in two ways: in the horizontal direction and the vertical direction. (Please make the drawing as you go along...) A horizontal line between $A(t)$ and $D(t)$ corresponds to the waiting time of a job, while a vertical line corresponds to the number of jobs in the system at time t. Now adding all horizontal lines (by integrating along the y-axis) makes up the total amount of waiting done by all the jobs until time T. On the other hand, adding the vertical lines (by integrating along the x-axis) is equal to the summation of all jobs in the system. Since the area is the same no matter whether you sum it in the horizontal or vertical direction:

$$\sum_{k=1}^{A(T)} W_k = \text{enclosed area} = \int_0^T (A(t) - D(t))\,dt.$$

Dividing both sides by $A(T)$ gives

$$\frac{1}{A(T)} \sum_{k=1}^{A(T)} W_k = \frac{1}{A(T)} \int_0^T (A(t) - D(t))\,dt.$$

Finally, observe that this equality holds between any two times T_i, T_{i+1}, where times $\{T_i\}$ are such that $A(T_i) = D(T_i)$. Then, as $T_i \to \infty$ which we assumed from the on-set, $\frac{1}{A(T_i)} \sum_{k=1}^{A(T_i)} W_k \to \mathsf{E}[W]$, and

$$\frac{T_i}{A(T_i)} \frac{1}{T_i} \int_0^{T_i} (A(t) - D(t))\,dt \to \lambda^{-1} \mathsf{E}[L].$$

Hence, Little's law follows.

s.1.13.5 Sometimes (often?) students memorize Little's law in the wrong way. Thus, as an easy check, use the dimensions of the concepts: $\mathsf{E}[L]$ is an average *number*, λ is a *rate*, i.e., *numbers per unit time*, and $\mathsf{E}[W]$ is waiting *time*.

s.1.13.6 Assume the system is rate-stable, for otherwise all relevant limits do not exists. The arrival rate at the server is λ and the time a job remains in at the server is $\mathsf{E}[S]$. Thus, the average time a job remains in the 'box' that identifies the server is $\mathsf{E}[S]$, and then the average number of jobs at the server is $\mathsf{E}[L_S] = \lambda \mathsf{E}[S]$, by Little's law. As $\lambda \mathsf{E}[S] = \rho$, we get $\mathsf{E}[L_S] = \rho$.

s.1.13.7 This was my initial answer (which is wrong): '$\mathsf{E}[W] = \lambda \mathsf{E}[L] = \lambda 10$'. Interestingly, I typed in Little's law in the wrong way... So, be aware! It's all too easy to make mistakes with Little's law.

This is correct:

$$\mathsf{E}[W] = \mathsf{E}[L]/\lambda = 10/\lambda = 10/5 = 2.$$

s.1.13.8 If you arrive at a queueing system, you first have to wait until the job in service is finished. Then you need to wait until the 9 jobs in queue are finished. This takes, in expectation, $9/\mu$. (Recall, 1 job is in service at the moment you arrive, so 9 are in queue.) Assuming that service times are exponential, so that, by the memoryless property, the remaining service time of the job in service is still $\mathsf{E}[S]$ when you arrive, you spend $10/\mu + 1/\mu = 11/6 \neq 2$. (To account for the last $+1/\mu$, observe that yourself also have to be served to compute the time you spend in the system.)

Now in this question, it is *given* that the system length is 10 at the moment of arrival. However, L as 'seen' upon arrival by this given customer is in general not the same as the time-average $\mathsf{E}[L]$.

Thus, Little's law need not hold at all moments in time; it is a statement about *averages*.

1.14 Applications and Useful Identities

Theory and Exercises

With the PASTA property and Little's law we can derive a number of useful and simple results for the $M/G/1$ queue by which we can analyze a large number of practical queueing situations, see Exercise 1.14.16 and below. In particular, we derive expressions for the following performance measures: server utilization, i.e., ρ, average queue length and waiting times. We assume that jobs arrive in accordance to a Poisson process so that we can use the PASTA property.

The fraction of time the server is empty is $1-\rho = p(0)$. By PASTA, $\pi(0) = p(0)$, hence the fraction of customers that enter an empty system is also $1-\rho$. Similarly, the fraction of customers that find the server occupied at arrival is equal to the the utilization ρ.

Suppose that at A_k, i.e., the arrival epoch of the kth job, the server is busy. The *remaining service time* $S_{r,k}$ as seen by job k is the time between A_k and the departure epoch of the job in service. If the server is free at A_k, we set $S_{r,k} = 0$. Define the average *remaining service time* as seen at arrival epochs as

$$\mathsf{E}[S_r] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} S_{r,k},$$

provided this limit exists.

1.14.1. It is an easy mistake to think that $\mathsf{E}[S_r] = \mathsf{E}[S]$ when service times are exponential. Why is this wrong?

1.14.2. ($M/G/1$) Use the PASTA property to show that

$$\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \,|\, S_r > 0]. \tag{1.66}$$

1.14.3. What would you guess for $\mathsf{E}[S_r \,|\, S_r > 0]$ for the $M/D/1$ queue?

1.14.4. Try to derive relation (1.66) with sample path arguments.

Next, consider the waiting time in queue.

1.14.5. ($M/G/1$) Show that the expected time in queue is

$$\mathsf{E}\left[W_Q\right] = \mathsf{E}[S_r] + \mathsf{E}\left[L_Q\right]\mathsf{E}[S]. \tag{1.67}$$

1.14.6. ($M/G/1$) Use the previous problem and Little's law to derive that

$$\mathsf{E}\left[W_Q\right] = \frac{\mathsf{E}[S_r]}{1-\rho} = \frac{\rho}{1-\rho}\,\mathsf{E}[S_r \,|\, S_r > 0]. \tag{1.68}$$

The average waiting time $\mathsf{E}[W]$ in the entire system, i.e., in queue plus in service, becomes

$$\mathsf{E}[W] = \mathsf{E}\left[W_Q\right] + \mathsf{E}[S] = \frac{\mathsf{E}[S_r]}{1-\rho} + \mathsf{E}[S].$$

The situation can be significantly simplified for the $M/M/1$ queue as then the service times are also exponential, hence memoryless, implying that $\mathsf{E}[S_r \,|\, S_r > 0] = \mathsf{E}[S]$.

1.14.7. ($M/M/1$) Derive that

$$\mathsf{E}[W] = \frac{\mathsf{E}[S]}{1-\rho}. \tag{1.69}$$

1.14.8. ($M/M/1$) Use the PASTA property to conclude that

$$\mathsf{E}[W] = \mathsf{E}[L]\,\mathsf{E}[S] + \mathsf{E}[S], \tag{1.70}$$

and apply Little's law to find (1.69).

1.14.9. Why is Eq. (1.70) not true in general for the $M/G/1$ queue?

1.14.10. ($M/M/1$) Show that

$$\mathsf{E}\left[W_Q\right] = \mathsf{E}[L]\,\mathsf{E}[S] = \frac{\rho^2}{1-\rho}\frac{1}{\lambda}, \tag{1.71}$$

Finally, we find expressions for the average lengths.

1.14.11. ($M/M/1$) Show that

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

1.14.12. ($M/M/1$) Explain that $\mathsf{E}\left[L_Q\right] = \sum_{n=1}^{\infty}(n-1)\pi(n)$, and use this to find that $\mathsf{E}\left[L_Q\right] = \rho^2/(1-\rho)$.

1.14.13. ($M/M/1$) Use the PASTA property to see that the expected waiting time in the system must also be equal to $\mathsf{E}[W] = \sum_{n=0}^{\infty}\mathsf{E}\left[W_Q \,\middle|\, N=n\right]\pi(n) + \mathsf{E}[S]$, where $\mathsf{E}\left[W_Q \,\middle|\, N=n\right]$ is the waiting time in queue given that an arrival sees $N = n$ customers upon arrival. Then, motivate that $\mathsf{E}\left[W_Q \,\middle|\, N=1\right] = \mathsf{E}[S]$. Finally, combine the above to see that $\mathsf{E}[W] = \mathsf{E}[S]\sum_{n=0}^{\infty} n\pi(n) + \mathsf{E}[S] = \mathsf{E}[S]\,\mathsf{E}[L] + \mathsf{E}[S]$.

1.14.14. ($M/M/1$) Show that the variance of the number of jobs in the system L is $\mathsf{V}[L] = \rho/(1-\rho)^2$.

1.14.15. ($M/M/1$) To see how large the variance of L is, relative to the mean number of jobs in the system, we typically consider the square coefficient of variation (SCV). Show that the square coefficient of variation of L is $1/\rho$. What do you conclude from this?

When a problem is mainly of a computational type, I coded the solutions and show you all the steps in between so that you can check each step in your computations. As the code is typically nearly identical to the mathematical formulas, you should not have any difficulty understanding the code. (In the computations below I typically use the simplest, but often not the most efficient, code.)

1.14.16. (Hall 5.2) After observing a single server queue for several days, the following steady-state probabilities have been determined: $p(0) = 0.4$, $p(1) = 0.3$, $p(2) = 0.2$, $p(3) = 0.05$ and $p(4) = 0.05$. The arrival rate is 10 customers per hour.

1. Determine $\mathsf{E}[L]$ and $\mathsf{E}\left[L_Q\right]$.
2. Using Little's formula, determine $\mathsf{E}[W]$ and $\mathsf{E}\left[W_Q\right]$.
3. Determine $\mathsf{V}[L]$ and $\mathsf{V}\left[L_Q\right]$.
4. Determine the service time and the utilization.

1.14.17. (Hall 5.5) An $M/M/1$ queue has an arrival rate of 100 per hour and a service rate of 140 per hour. What is $p(n)$? What are $\mathsf{E}\left[L_Q\right]$, $\mathsf{E}[L]$

1.14.18. (Hall, 5.6) An $M/M/1$ queue has been found to have an average waiting time in queue of 1 minute. The arrival rate is known to be 5 customers per minute. What are the service rate and utilization? Calculate $\mathsf{E}\left[L_Q\right]$, $\mathsf{E}[L]$ and $\mathsf{E}[W]$. Finally, the queue operator would like to provide chairs for waiting customers. He would like to have a sufficient number so that all customers can sit

down at least 90 percent of the time. How many chairs should he provide?

1.14.19. (Hall 5.7). A single server queueing system is known to have Poisson arrivals and exponential service times. However, the arrival rate and service time are state dependent. As the queue becomes longer, servers work faster, and the arrival rate declines, yielding the following functions (all in units of number per hour): $\lambda(0) = 5$, $\lambda(1) = 3$, $\lambda(2) = 2$, $\lambda(n) = 0, n \geq 3$, $\mu(0) = 0$, $\mu(1) = 2$, $\mu(2) = 3$, $\mu(n) = 4, n \geq 3$. Calculate the state probabilities, i.e., $p(n)$ for $n = 0, \ldots$.

1.14.20. (Hall 5.14) An airline phone reservation line has one server and a buffer for two customers. The arrival rate 6 customers per hour, and a service rate of just 5 customers per hour. Arrivals are Poisson and service times are exponential. Estimate $\mathsf{E}\left[L_Q\right]$ and the average number of customers served per hour. Then, estimate $\mathsf{E}\left[L_Q\right]$ for a buffer of size 5. What is the impact of the increased buffer size on the number of customers served per hour?

1.14.21. (Hall 5.3) After observing a queue with two servers for several days, the following steady-state probabilities have been determined: $p(0) = 0.4$, $p(1) = 0.3$, $p(2) = 0.2$, $p(3) = 0.05$ and $p(4) = 0.05$. The arrival rate is 10 customers per hour.

1. Determine $\mathsf{E}[L]$ and $\mathsf{E}\left[L_Q\right]$.
2. Using Little's formula, determine $\mathsf{E}[W]$ and $\mathsf{E}\left[W_Q\right]$.
3. Determine $\mathsf{V}[L]$ and $\mathsf{V}\left[L_Q\right]$.
4. Determine the service time and the utilization.

1.14.22. (Hall 5.8) The queueing system at a fast-food stand behaves in a peculiar fashion. When there is no one in the queue, people are reluctant to use the stand, fearing that the food is unsavory. People are also reluctant to use the stand when the queue is long. This yields the following arrival rates (in numbers per hour): $\lambda(0) = 10$, $\lambda(1) = 15$, $\lambda(2) = 15$, $\lambda(3) = 10$, $\lambda(4) = 5$, $\lambda(n) = 0, n \geq 5$. The stand has two servers, each of which can operate at 5 per hour. Service times are exponential, and the arrival process is Poisson. Calculate the steady

state probabilities. Next, what is the average arrival rate? Finally, Determine $\mathsf{E}[L]$, $\mathsf{E}[L_Q]$, $\mathsf{E}[W]$ and $\mathsf{E}[W_Q]$.

1.14.23. (Hall 5.10) A repair/maintenance facility would like to determine how many employees should be working in its tool crib. The service time is exponential, with mean 4 minutes, and customers arrive by a Poisson process with rate 28 per hour. The customers are actually maintenance workers at the facility, and are compensated at the same rate as the tool crib employees. What is $\mathsf{E}[W]$ for $c = 1, 2, 3$, or 4 servers? How many employees should work in the tool crib?

Hints

h.1.14.1 Realize again that $\mathsf{E}[S_r]$ includes the jobs that arrive at an empty system.

h.1.14.2 By the PASTA property, a fraction ρ of the arrivals sees the server occupied, while a fraction $1-\rho$ sees a free server.

h.1.14.4 This requires some extra definitions, similar to $A(n,t)$ as defined in (1.40a).

h.1.14.5 What happens when you enter a queue? First you have to wait until the job in service (if there is any) completes, and then you have to wait for the queue to clear.

h.1.14.6 Use Little's law to see that $\mathsf{E}[L_Q] = \lambda \mathsf{E}[W_Q]$. Substitute this in (1.67) and simplify.

h.1.14.8 Another way to derive the above result is to conclude from PASTA that the expected number of jobs in the system at an arrival is $\mathsf{E}[L]$. Since all these jobs require an expected service time $\mathsf{E}[S]$, including the job in service by the memoryless property, the time in queue is $\mathsf{E}[L]\,\mathsf{E}[S]$. The time in the system is then the waiting time plus the service time. Next, use Little's law $\mathsf{E}[L] = \lambda \mathsf{E}[W]$.

h.1.14.9 Think about the consequences of memoryless service times.

h.1.14.11 Apply Little's law to the $\mathsf{E}[L_Q]$ and so on, and use the earlier expressions for $\mathsf{E}[W_Q]$ and so on.

h.1.14.14 $\mathsf{V}[L] = \mathsf{E}[L^2] - (\mathsf{E}[L])^2$. We already know $\mathsf{E}[L]$ so it remains to compute $\mathsf{E}[L^2]$. Then, $\mathsf{E}[L^2] = \sum_{n=0}^{\infty} n^2 \pi(n)$. Use that $\pi(n) = \rho^n(1-\rho)$ is the probability to see n jobs in the system. Now simplify.

You might use the moment-generating trick of Exercise 1.10.1.

h.1.14.15 SCV = $\mathsf{V}[X]/(\mathsf{E}[X])^2$.

h.1.14.18 $\mathsf{E}\left[L_Q\right]$ follows right away from an application of Little's law. For the other quantities we need to find $\mathsf{E}[S]$. One can use that

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

Now λ and $\mathsf{E}\left[W_Q\right] = 1$ are given, and $\mathsf{E}\left[L_Q\right]$ has just been computed. Hence, $\mathsf{E}[S]$ (which is the unknown here) can computed with the abc-formula.

Another way is to realize that, for the $M/M/1$-queue, $\mathsf{E}\left[W_Q\right] = \frac{1}{\lambda}\frac{\rho^2}{1-\rho}$. Then solve for ρ, and since λ is known, $\mathsf{E}[S]$ follows.

h.1.14.19 Use the level-crossing equations of the $M(n)/M(n)/1$ queue.

h.1.14.20 This is a queueing system with loss, in particular the $M/M/1/1+2$ queue.

h.1.14.23 Realize that we have to control the number of servers. Hence, we are dealing with a multi-server queue, i.e., the $M/M/c$ queue. Use the formulas of Eq. (1.52).

The remark that maintenance workers are compensated at the same rate as the tool crib workers confused me a bit at first. Some thought revealed that the consequence of this remark is that is it just as expensive to let the tool crib workers wait (to help maintenance workers) as to let the maintenance workers wait for tools. (Recall, in queueing systems always somebody has to wait, either the customer in queue or the server being idle. If it is very expensive to let customers wait, the number of servers must be high, whereas if servers are relatively expensive, customers have to do the waiting.)

Solutions

s.1.14.1 $\mathsf{E}[S_r \,|\, S_r > 0] = \mathsf{E}[S]$ for the $M/M/1$ queue, and $\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \,|\, S_r > 0]$ for the $M/G/1$ queue, it follows that

$$\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \,|\, S_r > 0] = \rho\,\mathsf{E}[S].$$

s.1.14.2

$$\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \,|\, S_r > 0] + (1-\rho)\cdot\mathsf{E}[S_r \,|\, S_r = 0] = \rho\,\mathsf{E}[S_r \,|\, S_r > 0],$$

since, evidently, $\mathsf{E}[S_r \,|\, S_r = 0] = 0$.

s.1.14.3 Since the service times are deterministic (and constant), I would guess that on average half of the service time remains at the moment a job arrives. If the service time is D always, then $\mathsf{E}[S_r \,|\, S_r > 0] = \lambda D/2$.

s.1.14.4 This is, admittedly, not simple, so let us work in stages. Define $A(+,t)$ as the number of arrivals that see a job in service. As the server is only busy at time t when $L(t) > 0$, we define

$$A(+,t) = \sum_{k=1}^{\infty} \mathbb{1}[A_k \le t]\, \mathbb{1}[L(A_k-) > 0].$$

Next, define $\tilde{D}_k = \min\{D_i : D_i \ge A_k\}$ as the first departure after the arrival time A_k of job k. We need to consider two cases. If $\tilde{D}_k = D_k$ then, obviously, the first departure after A_k coincides with the departure time of job k. This is only possible if $L(A_k-) = 0$. But then, it must be that the remaining service time of the job in service at time A_k- is 0, as there is no job in service. Otherwise, if $L(A_k-) > 0$ it must be that $\tilde{D}_k < D_k$, and, as a consequence, the remaining service time of the job in service at time A_k- is equal to $\tilde{D}_k - A_k$. All in all, the total amount of remaining service times added up for all arrivals up to time t can be written as

$$S_r(t) = \sum_{k=1}^{\infty} (\tilde{D}_k - A_k)\, \mathbb{1}[A_k \le t, L(A_k-) > 0].$$

The remaining service time averaged over all arrivals up to time t is therefore $S_r(t)/A(t)$. We can now rewrite this to

$$\frac{S_r(t)}{A(t)} = \frac{A(+,t)}{A(t)} \frac{S_r(t)}{A(+,t)},$$

and interpret these fractions. First, consider

$$\frac{A(+,t)}{A(t)} = \frac{\sum_{k=1}^{\infty} \mathbb{1}[A_k \le t, L(A_k-) > 0]}{\sum_{k=1}^{\infty} \mathbb{1}[A_k \le t]}.$$

This is the number of jobs up to time t that see at least one job in the system divided by the total number of arrivals up to time t. Thus, as $t \to \infty$,

$$\frac{A(+,t)}{A(t)} \to \sum_{n=1}^{\infty} \pi(n) = 1 - \pi(0).$$

With PASTA we can conclude that $1 - \pi(0) = 1 - p(0) = \rho$. Second,

$$\frac{S_r(t)}{A(+,t)} = \frac{\sum_{k=1}^{\infty} (\tilde{D}_k - A_k)\, \mathbb{1}[A_k \le t, L(A_k-) > 0]}{\sum_{k=1}^{\infty} \mathbb{1}[A_k \le t, L(A_k-) > 0]},$$

is the total amount of remaining service time up to time t divided by the number of arrivals up to time t that see at least one job in the system. Thus, if the limit exists, we can *define*

$$\mathsf{E}[S_r \,|\, S_r > 0] = \lim_{t\to\infty} \frac{S_r(t)}{A(+,t)}.$$

s.1.14.5 It is evident that the expected waiting time for an arriving customer is the expected remaining service time plus the expected time in queue. The expected time in queue must be equal to the expected number of customers in queue at an arrival epoch times the expected service time per customer, assuming that service times are i.i.d. If the arrival process is Poisson, it follows from PASTA that the average number of jobs in queue perceived by arriving customers is also the *time-average* number of jobs in queue $\mathsf{E}\left[L_Q\right]$.

s.1.14.6 With Little's law $\mathsf{E}\left[L_Q\right] = \lambda\,\mathsf{E}\left[W_Q\right]$. Using this,

$$\mathsf{E}\left[W_Q\right] = \mathsf{E}[S_r] + \lambda\,\mathsf{E}\left[W_Q\right]\mathsf{E}[S] = \mathsf{E}[S_r] + \rho\,\mathsf{E}\left[W_Q\right],$$

since $\rho = \lambda\,\mathsf{E}[S]$. But this gives for the $M/G/1$ queue that

$$\mathsf{E}\left[W_Q\right] = \frac{\mathsf{E}[S_r]}{1-\rho} = \frac{\rho}{1-\rho}\,\mathsf{E}[S_r \,|\, S_r > 0],$$

where we use (1.66) in the last equation.

s.1.14.7 Using (1.66),

$$\begin{aligned}\mathsf{E}[W] &= \mathsf{E}\left[W_Q\right] + \mathsf{E}[S] = \frac{\mathsf{E}[S_r]}{1-\rho} + \mathsf{E}[S] \\ &= \frac{\rho\,\mathsf{E}[S]}{1-\rho} + \mathsf{E}[S] = \frac{\mathsf{E}[S]}{1-\rho}.\end{aligned}$$

s.1.14.8 From the hint

$$\mathsf{E}[W] = \mathsf{E}[L]\,\mathsf{E}[S] + \mathsf{E}[S] = \lambda\,\mathsf{E}[W]\,\mathsf{E}[S] + \mathsf{E}[S] = \rho\,\mathsf{E}[W] + \mathsf{E}[S]. \tag{1.72}$$

Substituting Little's law $\mathsf{E}[L] = \lambda\,\mathsf{E}[W]$ and simplifying,

$$\mathsf{E}[W] = \frac{\mathsf{E}[S]}{1-\rho}.$$

s.1.14.9 Because the remaining service time of the job in service, provided there is a job in the system upon arrival, is not exponentially distributed in general. Only for the $M/M/1$ queue the service times are exponentially distributed, hence memoryless. And only when the service time is memoryless, the service time after an interruption is still exponential.

s.1.14.10

$$\mathsf{E}\left[W_Q\right] = \mathsf{E}[L]\,\mathsf{E}[S] = \lambda\,\mathsf{E}[W]\,\mathsf{E}[S] = \rho\,\mathsf{E}[W] = \frac{\rho}{1-\rho}\,\mathsf{E}[S] = \frac{\rho^2}{1-\rho}\frac{1}{\lambda}, \tag{1.73}$$

which is consistent with our earlier result.

s.1.14.11 With Little's law,

$$\begin{aligned}
\mathsf{E}[L] &= \lambda\,\mathsf{E}[W] = \frac{\lambda\,\mathsf{E}[S]}{1-\rho} = \frac{\rho}{1-\rho},\\
\mathsf{E}\left[L_Q\right] &= \lambda\,\mathsf{E}\left[W_Q\right] = \frac{\rho^2}{1-\rho},\\
\mathsf{E}[L_s] &= \mathsf{E}[L] - \mathsf{E}\left[L_Q\right] = \frac{\rho}{1-\rho} - \frac{\rho^2}{1-\rho} = \rho,
\end{aligned}$$

since, the expected number of jobs in service $\mathsf{E}[L_s]$ is equal to the expected number of busy servers.

s.1.14.12 The fraction of time the system contains n jobs is $\pi(n)$ (by PASTA). When the system contains $n > 0$ jobs, the number in queue is one less, i.e., $n-1$.

$$\begin{aligned}
\mathsf{E}\left[L_Q\right] &= \sum_{n=1}^{\infty}(n-1)\pi(n) = (1-\rho)\sum_{n=1}^{\infty}(n-1)\rho^n\\
&= \rho(1-\rho)\sum_{n=1}^{\infty}(n-1)\rho^{n-1} = \rho\sum_{n=1}^{\infty}(n-1)\pi(n-1)\\
&= \rho\sum_{n=0}^{\infty}n\pi(n) = \rho\frac{\rho}{1-\rho}.
\end{aligned}$$

Another way to get the same result is by splitting:

$$\begin{aligned}
\mathsf{E}\left[L_Q\right] &= \sum_{n=1}^{\infty}(n-1)\pi(n) = \sum_{n=1}^{\infty}n\pi(n) - \sum_{n=1}^{\infty}\pi(n)\\
&= \mathsf{E}[L] - (1-\pi(0)) = \mathsf{E}[L] - \rho.
\end{aligned}$$

s.1.14.13 1. The time in the system $\mathsf{E}[W]$ is the sum of the time in queue plus the service time of the job itself. Conditioning on the number of jobs N in the system at arrival moments, i.e., conditioning on $\pi(n)$, gives the result.

2. $\mathsf{E}\left[W_Q \,\middle|\, N=0\right] = 0$, of course. If $N=1$, there must be a job in service. Because of the memoryless property of the service times, the remaining service time of the customer in service is still exponentially distributed with mean $\mathsf{E}[S]$. Thus, $\mathsf{E}\left[W_Q \,\middle|\, N=1\right] = \mathsf{E}[S]$.

3. The expected service time for a customer still in queue is $\mathsf{E}[S]$. Therefore, for each n we have that, when service times are exponential, $\mathsf{E}\left[W_Q \,\middle|\, N=n\right] = n\,\mathsf{E}[S]$. The probability to see n jobs in the system by an arrival is $\pi(n)$ which, by PASTA, is equal to $p(n)$. Since $\mathsf{E}[L] = \sum_n np(n)$, the result follows.

s.1.14.14 The starting point is $\mathsf{V}[L] = \mathsf{E}\left[L^2\right] - (\mathsf{E}[L])^2$.
With wolfram alpha we get

$$\begin{aligned}\mathsf{E}\left[L^2\right] &= \sum_{n=0}^{\infty} n^2\pi(n) \\ &= \rho\frac{1+\rho}{(1-\rho)^2}.\end{aligned}$$

Thus,

$$\mathsf{V}[L] = \frac{\rho(1+\rho)}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} = \frac{\rho}{(1-\rho)^2}.$$

Here are three methods to to get the result for $\mathsf{E}\left[L^2\right]$.
One way is to use the standard formula for a geometric series with $\rho < 1$:

$$\frac{1}{1-\rho} = \sum_{n=0}^{\infty} \rho^n.$$

If we differentiate the left and right hand side with respect to ρ and then multiply with ρ we obtain

$$\frac{\rho}{(1-\rho)^2} = \sum_{n=0}^{\infty} n\rho^n.$$

Again, differentiating and multiplying with ρ yields,

$$\begin{aligned}\rho\frac{(1-\rho)^2+\rho 2(1-\rho)}{(1-\rho)^4} &= \rho\frac{1-2\rho+\rho^2+2\rho-2\rho^2}{(1-\rho)^4}\\ &= \rho\frac{(1-\rho)^2}{(1-\rho)^4}\\ &= \rho\frac{1+\rho}{(1-\rho)^3}\\ &= \sum_{n=0}^{\infty} n^2\rho^n\end{aligned}$$

and hence

$$(1-\rho)\sum_{n=0}^{\infty} n^2\rho^n = \rho\frac{1+\rho}{(1-\rho)^2}$$

Observe that here we just compute the second moment of a geometric random variable.

Another way to derive the result is by noting that $\sum_{i=1}^{n} i = n(n+1)/2$ from which we get that $n^2 = -n + 2\sum_{i=1}^{n} i$. Substituting this relation into $\sum_n n^2\rho^n$ leads to

$$\begin{aligned}\sum_{n=0}^{\infty} n^2\rho^n &= \sum_{n=0}^{\infty}\left(\sum_{i=1}^{\infty} 2i\,\mathbb{1}[i\le n]-n\right)\rho^n = \sum_{n=0}^{\infty}\sum_{i=0}^{\infty} 2i\,\mathbb{1}[i\le n]\rho^n - \sum_{n=0}^{\infty} n\rho^n\\ &= \sum_{i=0}^{\infty} 2i\sum_{n=i}^{\infty}\rho^n - \frac{\mathsf{E}[L]}{1-\rho} = \sum_{i=0}^{\infty} 2i\rho^i\sum_{n=0}^{\infty}\rho^n - \frac{\mathsf{E}[L]}{1-\rho}\\ &= \frac{2}{1-\rho}\sum_{i=0}^{\infty} i\rho^i - \frac{\mathsf{E}[L]}{1-\rho} = \frac{2}{(1-\rho)^2}\mathsf{E}[L] - \frac{\mathsf{E}[L]}{1-\rho}\\ &= \frac{\mathsf{E}[L]}{1-\rho}\left(\frac{2}{1-\rho}-1\right) = \frac{\mathsf{E}[L]}{1-\rho}\frac{1+\rho}{1-\rho}\\ &= \frac{\rho}{1-\rho}\frac{1+\rho}{(1-\rho)^2}.\end{aligned}$$

A third method is based on z-transforms, which is the same thing as moment generating functions. Writing $z = e^s$, we have $\mathsf{E}\left[e^{sL}\right] = \mathsf{E}\left[z^L\right]$. Hence

$$\phi(z) = \mathsf{E}\left[z^L\right] = \sum_{n=0}^{\infty} z^n p(n) = (1-\rho)\sum_{n=0}^{\infty}(\rho z)^n = \frac{1-\rho}{1-\rho z}.$$

Then

$$\mathsf{E}[L] = \frac{d}{dz}\phi(z)\bigg|_{z=1} = \frac{\rho}{1-\rho},$$

and

$$\mathsf{E}[L(L-1)] = \phi''(z)\big|_{z=1} = \frac{2\rho^2}{(1-\rho)^2}.$$

Thus,

$$\mathsf{E}[L]^2 = \mathsf{E}[L(L-1)] + \mathsf{E}[L].$$

Hence,

$$\begin{aligned}\mathsf{E}[L]^2 &= \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho} \\ &= \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{(}1-\rho)(1-\rho)2 \\ &= \frac{\rho^2+\rho}{(1-\rho)^2} = \rho\frac{1+\rho}{(1-\rho)^2}.\end{aligned}$$

A bit of algebra gives the previous results.

s.1.14.15 To see how large this variance is, relative to the mean number of jobs in the system, we typically consider the square coefficient of variation (SCV). As $\mathsf{E}[L] = \rho/(1-\rho)$,

$$\frac{\mathsf{V}[L]}{(\mathsf{E}[L])^2} = \frac{1}{\rho}.$$

Thus, the SCV becomes smaller as ρ increases, but does not become lower than 1. So, realizing that the SCV of the exponential distribution is 1, the distribution of the number of jobs in the system has larger relative variability than the exponential distribution.

s.1.14.16 First find $\mathsf{E}[L]$

```
>>> P = [0.4, 0.3, 0.2, 0.05, 0.05]
>>> L = sum(n*P[n] for n in range(len(P)))
>>> L
1.05
```

There can only be a queue when a job is in service. Since there is $m = 1$ server, we subtract m from the amount of jobs in the system. Before we do this, we need to ensure that $n - m$ does not become negative. Thus, $\mathsf{E}[L_Q] = \sum_n \max\{n-m, 0\}p(n)$.

```
>>> m = 1
>>> Lq = sum(max(n-m,0)*P[n] for n in range(len(P)))
>>> Lq
0.45000000000000007
```

```
>>> labda = 10./60
>>> Wq = Lq/labda # in minutes
>>> Wq
2.7000000000000006
>>> Wq/60 # in hours
0.04500000000000001

>>> W = L/labda # in minutes
>>> W
6.300000000000001
>>> W/60 # in hours
0.10500000000000001

>>> from math import sqrt
>>> var_L = sum((n-L)**2*P[n] for n in range(len(P)))
>>> var_L
1.2475
>>> sqrt(var_L)
1.116915395184434

>>> var_Lq = sum((max(n-m,0)-Lq)**2*P[n] for n in range(len(P)))
>>> var_Lq
0.6475
>>> sqrt(var_Lq)
0.8046738469715541

>>> mu = 1./(W-Wq)
>>> 1./mu # in minutes
3.5999999999999996

>>> rho = labda/mu
>>> rho
0.6

>>> rho = L-Lq
>>> rho
0.6
```

This checks the previous line.

The utilization must also by equal to the fraction of time the server is busy.

```
>>> u = 1 - P[0]
>>> u
0.6
```

Yet another way: Suppose we have m servers. If the system is empty, all m servers are idle. If the system contains one customer, $m-1$ servers are idle. Therefore, in general, the average fraction of time the server is idle is

$$1-u=\sum_{n=0}^{\infty}\max\{n-m,0\}p_n,$$

as in the case there are more than m customers in the system, the number of idle servers is 0.

```
>>> idle = sum( max(m-n,0)*P[n] for n in range(len(P)))
>>> idle
0.4
```

s.1.14.17 1. $p(n)=(1-\rho)\rho^n$

2.
```
>>> labda = 100. # per hour
>>> mu = 140. # per hour
>>> ES = 1./mu
>>> rho = labda/mu
>>> rho
0.7142857142857143
>>> 1-rho
0.2857142857142857

>>> L = rho/(1.-rho)
>>> L
2.5
>>> Lq = rho**2/(1.-rho)
>>> Lq
1.7857142857142858

>>> W = 1./(1.-rho) * ES
>>> W
0.024999999999999998
>>> Wq = rho/(1.-rho) * ES
>>> Wq
0.017857142857142856
```

s.1.14.18 $\mathsf{E}[W_Q] = \frac{1}{\lambda}\frac{\rho^2}{1-\rho}$. Since $\mathsf{E}[W_Q]$ and λ is given we can use this formula to solve for ρ with the abc-formula (and using that $\rho > 0$):

```
>>> labda = 5. # per minute
>>> Wq = 1.
>>> a = 1.
>>> b = labda*Wq
>>> c = -labda*Wq
>>> rho = (-b + sqrt(b*b-4*a*c))/(2*a)
>>> rho
0.8541019662496847

>>> ES = rho/labda
>>> ES
0.17082039324993695

>>> Lq = labda*Wq
>>> Lq
5.0

>>> W = Wq + ES
>>> W
1.170820393249937

>>> L = labda*W
>>> L
5.854101966249685
```

The next problem is to find n such that $\sum_{j=0}^{n} p_j > 0.9$.

```
>>> total = 0.
>>> j = 0
>>> while total <= 0.9:
...     total += (1-rho)*rho**j
...     j += 1
...
>>> total
0.9061042738330157
>>> j
```

```
15
>>> n = j- 1 # the number of chairs
```

Observe that j is one too high once the condition is satisfied, thus subtract one. As a check, I use that $(1-\rho)\sum_{j=0}^{n}\rho^j = 1-\rho^{n+1}$.

```
>>> 1-rho**(n) #  this must be too small.
0.890064968964683
>>> 1-rho**(n+1) # this must be OK.
0.9061042738330156
```

And indeed, we found the right n.

s.1.14.19 Follows right away from the hint.

s.1.14.20 1. First compute $\mathsf{E}[L_Q]$ for the case with a buffer for 2 customers.

```
>>> labda = 6.
>>> mu = 5.
>>> rho = labda/mu
>>> c = 1
>>> b = 2
```

Set $p(n) = \rho^n$ initially, and normalize later. Use the expressions for the $M(n)/M(n)$ queue. Observe that $\rho > 1$. Since the size of the system is $c+b+1$ is finite, all formulas work for this case too.

There are 4 states in total: $0,1,2,3$. (The reason to import numpy here and convert the lists to arrays is to fix the output precision to 3, otherwise we get long floats in the output.)

```
>>> import numpy as np
>>> np.set_printoptions(precision=3)

>>> P = np.array([rho**n for n in range(c+b+1)])
>>> P
array([1.   , 1.2  , 1.44 , 1.728])

>>> G = sum(P)
>>> G
5.368
```

```
>>> P /= G # normalize
>>> P
array([0.186, 0.224, 0.268, 0.322])

>>> L = sum(n*P[n] for n in range(len(P)))
>>> L
1.725782414307004

>>> Lq = sum((n-c)*P[n] for n in range(c,len(P)))
>>> Lq
0.9120715350223545
```

The number of jobs served per hour must be equal to the number of jobs accepted, i.e., not lost. The fraction of customers lost is equal to the fraction of customers that sees a full system.

```
>>> lost = labda*P[-1] # the last element of P
>>> lost
1.9314456035767507

>>> accepted = labda*(1.-P[-1]) # rate at which jobs are accepted
>>> accepted
4.06855439642325
```

2. Now increase the buffer b to 5.

```
>>> b = 5
>>> P = np.array([rho**n for n in range(c+b+1)])
>>> P
array([1.   , 1.2  , 1.44 , 1.728, 2.074, 2.488, 2.986])
>>> G = sum(P)
>>> G
12.915903999999998

>>> P /= G # normalize
>>> P
array([0.077, 0.093, 0.111, 0.134, 0.161, 0.193, 0.231])

>>> L = sum(n*P[n] for n in range(len(P)))
>>> L
3.7098374221424995
```

```
>>> accepted = labda*(1.-P[-1])
>>> accepted
4.6128803682653565
```

s.1.14.21 1. Determine $\mathsf{E}[L]$ and $\mathsf{E}\left[L_Q\right]$.

```
>>> P = [0.4, 0.3, 0.2, 0.05, 0.05]

>>> c = 2
>>> Lq = sum((n-c)*P[n] for n in range(c,len(P)))
>>> Lq
0.15000000000000002

>>> L= sum(n*P[n] for n in range(len(P)))
>>> L
1.05
```

2. Using Little's formula, determine $\mathsf{E}[W]$ and $\mathsf{E}\left[W_Q\right]$.

```
>>> labda = 10./60
>>> Wq = Lq/labda # in minutes
>>> Wq
0.9000000000000001
>>> Wq/60 # in hours
0.015000000000000003

>>> W = L/labda
>>> W
6.300000000000001
```

3. Determine $\mathsf{V}[L]$ and $\mathsf{V}\left[L_Q\right]$.

```
>>> from math import sqrt
>>> var_L = sum((n-L)**2*P[n] for n in range(len(P)))
>>> var_L
1.2475
>>> sqrt(var_L)
1.116915395184434
```

4. Determine the service time and the utilization.

```
>>> var_q = sum((max(n-c,0)-Lq)**2*P[n] for n in range(len(P)))
>>> var_q
0.22750000000000004
>>> sqrt(var_q)
0.4769696007084729
```

s.1.14.22 First the service rates.

```
>>> import numpy as np
>>> labda = [10., 15., 15., 10., 5.]
>>> c = 2
>>> mn = 2*np.ones(len(labda)+1, dtype=int)  # number of active servers
>>> mn[0] = 0  # no service if system is empty
>>> mn[1] = 1  # one busy server if just one job present
>>> mu = 5*mn # service rate is 5 times no of active servers
>>> mu
array([ 0,  5, 10, 10, 10, 10])
```

Since there can be arrivals in states $0,\ldots,4$, the system can contain 0 to 5 customers, i.e., $p(0),\ldots,p(5)$.

Use the level crossing result for the $M(n)/M(n)/1$ queue:

```
>>> P = [1]*(len(labda)+1)
>>> for i in range(1,len(P)):
...     P[i] = labda[i-1]/mu[i]*P[i-1]
...
>>> P = np.array(P) # unnormalized probabilities
>>> P
array([1.  , 2.  , 3.  , 4.5 , 4.5 , 2.25])
```

```
>>> G = sum(P) # normalization constant
>>> G
17.25
>>> P /= G # normalize
>>> P
array([0.058, 0.116, 0.174, 0.261, 0.261, 0.13 ])
```

$\lambda = \sum_n \lambda(n)p(n)$.

```
>>> labdaBar = sum(labda[n]*P[n] for n in range(len(labda)))
>>> labdaBar
8.840579710144928
```

The average number in the system is:

```
>>> Ls = sum(n*P[n] for n in range(len(P)))
>>> Ls
2.942028985507246
```

The average number in queue

```
>>> c = 2
>>> Lq = sum((n-c)*P[n] for n in range(c,len(P)))
>>> Lq
1.1739130434782608
```

And now the waiting times:

```
>>> Ws = Ls/labdaBar
>>> Ws # time in the system
0.3327868852459016

>>> Wq = Lq/labdaBar
>>> Wq # time in queue
0.13278688524590163
```

s.1.14.23 1. Would one server/person do?

```
>>> labda = 28./60 # arrivals per minute
>>> ES = 4.
>>> labda*ES
1.8666666666666667
```

If $c = 1$, the load $\rho = \lambda \mathsf{E}[S]/c > 1$ is clearly undesirable for one server. We need at least two servers.

It is not relevant to focus on the time in the system , as time in service needs to be spent anyway. Hence, we focus on the waiting time in queue.

I just convert the formulas of (1.52) to python code. This saves me time during the computations.

```
>>> def WQ(c, labda, ES):
...     rho = labda*ES/c
...     G = sum([(c*rho)**n/factorial(n) for n in range(c)])
...     G += (c*rho)**c/(1.-rho)/factorial(c)
```

```
...     Lq = (c*rho)**c/(factorial(c)*G) * rho/(1.-rho)**2
...     return Lq/labda # Wq, Little's law
...
```

Considering the scenario with one server is superfluous as $\rho > 1$ in that case.

What is the waiting time for $c = 2$ servers?

```
>>> WQ(2, 28./60, 4) # in minutes
27.034482758620694
>>> WQ(2, 28./60, 4)/60. # in hours
0.4505747126436782
```

What is the waiting time for $c = 3$ servers?

```
>>> WQ(3, 28./60, 4)   # in minutes
1.3542675591474136
>>> WQ(3, 28./60, 4)/60. # in hours
0.022571125985790228
```

What is the waiting time for $c = 4$ servers?

```
>>> WQ(4, 28./60, 4) # in minutes
0.26778942672317635
>>> WQ(4, 28./60, 4)/60. # in hours
0.0044631571120529396
```

In the next part of the question we will interpret these numbers.

2. Since both types of workers cost the same amount of money per unit time, it is best to divide the amount of waiting/idleness equally over both types of workers. I am inclined to reason as follows. The average amount of waiting time done by the maintenance workers per hour is $\lambda \mathsf{E}[W_Q]$. To see this, note that maintenance workers arrive at rate λ, and each worker waits on average $\mathsf{E}[W_Q]$ minutes. Thus, worker time is wasted at rate $\lambda \mathsf{E}[W_Q]$. Intestingly, with Little's lawy, $\mathsf{E}[L_Q] = \lambda \mathsf{E}[W_Q]$, i.e., the rate at which workers waste capacity (i.e. waiting in queue) is $\mathsf{E}[L_Q]$. On the other hand, the rate of work capacity wasted by the tool crib employees being idle is $c - \lambda \mathsf{E}[S]$, as $\lambda \mathsf{E}[S]$ is the average number of servers busy, while c crib servers are available.

   As both types of employees are equally expensive, we need to choose c such that the number of maintenance workers waiting (i.e., being idle because they are waiting in queue), is equal to the number of crib workers being idle. In

other words, we search for a c such that $\mathsf{E}[L_Q] \approx c - \lambda\,\mathsf{E}[S]$ (where, of course, $\mathsf{E}[L_Q]$ depends on c.)

```
>>> labda = 28./60
>>> ES = 4.
>>> c = 2
>>> ELQ = labda*WQ(c, labda, ES)
>>> ELQ
12.616091954022991
>>> c-labda*ES
0.1333333333333333
```

Now the maintenance employees wait more than the tool crib employees.

```
>>> c = 3
>>> ELQ = labda*WQ(c, labda, ES)
>>> ELQ
0.6319915276021264
>>> c-labda*ES
1.1333333333333333
```

```
>>> c = 4
>>> ELQ = labda*WQ(c, labda, ES)
>>> ELQ
0.1249683991374823
>>> c-labda*ES
2.1333333333333333
```

Clearly, $c = 3$ should do.

1.15 $\mathbf{M^X/M/1}$ Queue: Expected Waiting Time

Theory and Exercises

It is not always the case that jobs arrive in single units, they can also arrive as batches. For instance, when a car and or bus arrives at a fast food restaurant, a batch consists of the number of people in the vehicle. In this section we derive for such queueing processes, denoted by the shorthand $M^X/M/1$, expressions for the load and the expected waiting time and queue length.

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, i.e., the number of items that job k brings into the system. We assume that $\{B_k\}$ is a sequence of i.i.d. discrete random variables distributed as a generic random variable B such that $\mathsf{P}(B = k) = f(k)$, where $f(k)$ is a given set of probabilities. We write

$$G(k) = \mathsf{P}(B > k) = \sum_{m=k+1}^{\infty} f(m),$$

for the *survivor function* of B. We also assume that the service time of each item is exponentially distributed with average $1/\mu$. Thus, the average time to serve the entire batch is

$$\mathsf{E}[B]\,\mathsf{E}[S] = \mathsf{E}[B]/\mu.$$

The first criterion we must check for the $M^X/M/1$ queue is the stability: the service rate must be larger than the arrival rate of work. To determine the latter, observe that the total number of items $N(t)$ arrived up to time t must be equal to the number of arrivals $A(t)$ up to time t times the batch size of each arrival, i.e.,

$$N(t) = \sum_{k=1}^{\infty} B_k \, \mathbb{1}[A_k \le t] = \sum_{k=1}^{A(t)} B_k.$$

The (stochastic) process $\{N(t)\}$ is known as the *compound Poisson process*. Clearly, the rate at which items arrive is approximately

$$\frac{N(t)}{t} = \frac{A(t)}{t} \frac{1}{A(t)} \sum_{i=k}^{A(t)} B_k.$$

As in the limit $A(t)/t \to \lambda$ and $(A(t))^{-1} \sum_{k=1}^{A(t)} B_k \to \mathsf{E}[B]$, we see that

$$\frac{N(t)}{t} \to \lambda\,\mathsf{E}[B].$$

1.15.1. Conclude that the *load* must satisfy $\rho = \lambda\,\mathsf{E}[B]/\mu < 1$.

Let us next find expressions for the waiting time and queue lengths. For this, we need to describe the queueing process is somewhat more detail. A batch joins the end of the queue (if present), and once the queue in front of it is cleared, the entire batch moves from the queue to the server. Then the server starts serving the batch. As the service of the first item of the batch starts right away, this item does not have to wait. The last item, however, must wait for all the other items of the batch to

be served before its service can start. Consequently, all items of a batch, except the first, also have to wait at the server before service starts.

For the average waiting time, we use the derivation of $\mathsf{E}\left[W_Q\right] = \mathsf{E}[S_r]/(1-\rho)$, i.e., (1.68), as guidance. Suppose a batch finds $\mathsf{E}[L]$ items in the system upon arrival. By the memoryless property of the service distribution, the expected remaining service time of the item in service (if any) is $\mathsf{E}[S]$. Therefore the expected time the entire batch spends in queue is

$$\mathsf{E}\left[W_{Q,b}\right] = \mathsf{E}[L]\,\mathsf{E}[S];$$

compare Eq. 1.71. Next, if B_r is the number of items of the batch currently at the server ($B_r = 0$ if the server is idle), and $L_{Q,b}$ the number of batches in queue, then

$$\mathsf{E}[L] = \mathsf{E}\left[L_{Q,b}\right]\mathsf{E}[B] + \mathsf{E}[B_r].$$

1.15.2. Combine the above with Little's law to show that

$$\mathsf{E}\left[W_{Q,b}\right] = \frac{\mathsf{E}[B_r]}{1-\rho}\,\mathsf{E}[S].$$

Below we prove that the expected number of items at the server is given by

$$\mathsf{E}[B_r] = \frac{1}{2}\frac{\mathsf{E}\left[B^2\right]}{\mathsf{E}[B]}\rho + \frac{\rho}{2}. \tag{1.74}$$

1.15.3. To simplify this expression for $\mathsf{E}[B_r]$, show that

$$\frac{\mathsf{E}\left[B^2\right]}{\mathsf{E}[B]} = (1+C_s^2)\,\mathsf{E}[B], \quad \text{where } C_s^2 = \frac{\mathsf{V}[B]}{(\mathsf{E}[B])^2},$$

is the square coefficient of variation of the batch sizes.

With the results of the above exercises we can establish two cornerstones of queueing theory. The first is the expected waiting time,

$$\mathsf{E}\left[W_{Q,b}\right] = \frac{\mathsf{E}[B_r]}{1-\rho}\,\mathsf{E}[S] = \frac{1+C_s^2}{2}\frac{\rho}{1-\rho}\,\mathsf{E}[B]\,\mathsf{E}[S] + \frac{1}{2}\frac{\rho}{1-\rho}\,\mathsf{E}[S]. \tag{1.75}$$

Second, for the number of items in the system we find

$$\mathsf{E}[L] = \frac{\mathsf{E}\left[W_{Q,b}\right]}{\mathsf{E}[S]} = \frac{1+C_s^2}{2}\frac{\rho}{1-\rho}\,\mathsf{E}[B] + \frac{1}{2}\frac{\rho}{1-\rho}. \tag{1.76}$$

Thus, to compute the average number of items in the system, we only need to know the first and second moment (or the variance) of the batch size B. Thus, no matter how 'complicated' the distribution of B, when its second moment exists, the average queue length and waiting time can be computed with the above result.

1.15.4. Show that when the batch size is 1, the expression $\mathsf{E}\left[L(M^X/M/1)\right]$, i.e., the system length for the $M^X/M/1$ queue, reduces to $\mathsf{E}[L(M/M/1)]$, i.e., the system length for the $M/M/1$ queue. (Realize the importance of such checks.)

1.15.5. What is $\mathsf{E}[L]$ in case $B_k = 3$ always, and $\lambda = 1$, $\mu = 6$?

1.15.6. If the batch size is p geometrically distributed, what is $\mathsf{E}[L]$?

1.15.7. 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 for an item is 25 minutes. By how much would the waiting time decrease if batch sizes are constant and equal to 2 (observe that in both cases $\mathsf{E}[B] = 2$).

Observe how easy it is with these models to get insight into the order of magnitude of queue length reductions or waiting times that can be achieved with changing work habits, such making batch sizes constant rather than allowing them to vary. Observe also that it is up to management to decide whether such reductions outweigh any efforts to reduce the variation in batch sizes.

1.15.8. Show that $\mathsf{E}\left[L(M^X/M/1)\right] \geq \mathsf{E}[L(M/M/1)]$ when the loads are the same. What do you conclude?

1.15.9. In a production environment, a machine replenishes an inventory of items (e.g., hamburgers) at a fixed rate of 1 per 3 minutes. If the inventory reaches the *produce-up-to* level S, the machine stops. Customers arrive at rate of 6 per hour. A customer can buy items in different quantities, $B = 1,2,3,4$, all with equal probability. What is a sensible value for the produce-up-to level S?

We now turn to proving (1.74) with sample-path arguments and counting; it is an elegant line of reasoning. Moreover, this derivation of the distribution of the *remaining lifetime distribution* or the *residual life* is useful in its own right and and used in, for instance, maintenance planning.

For this purpose, consider the start and finish times of the batches in service. Then, remove all idle periods of the server, so that we paste together the times during which the server is busy, and plot the remaining job size as a function of time, c.f., Figure 1.19. Define R as the remaining number of items to be served of the batch in service at some arbitrary point in time. Let us show with Figure 1.19 that

$$\mathsf{P}(R = i) = \frac{\mathsf{P}(B \geq i)}{\mathsf{E}[B]} = \frac{G(i-1)}{\mathsf{E}[B]}.$$

In Figure 1.19 concentrate on level i. Only jobs whose initial batch size is larger or equal to i can produce i items at the server. Thus, by counting, we see in Figure 1.19 that $\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]$ is the number of times there are precisely i items at the server. We also see that $\sum_{k=1}^{n} B_k$ is the total time the server is busy. Thus, the fraction of time there are i remaining items is

$$\frac{\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]}{\sum_{k=1}^{n} B_k}.$$

By PASTA this is also the fraction of jobs that see i items at the server. Finally,

$$\mathsf{P}(R = i) = \lim_{n\to\infty} \frac{\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]}{\sum_{k=1}^{n} B_k} = \lim_{n\to\infty} \frac{n^{-1}\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]}{n^{-1}\sum_{k=1}^{n} B_k} = \frac{G(i-1)}{\mathsf{E}[B]},$$

where the limits exist by the strong law. With Exercise 1.15.10 the expected remaining time can be simplified to

$$\mathsf{E}[R] = \sum_{i=1}^{\infty} i\mathsf{P}(R = i) = \frac{1}{\mathsf{E}[B]} \sum_{i=1}^{\infty} iG(i-1) = \frac{\mathsf{E}[B^2]}{2\mathsf{E}[B]} + \frac{1}{2}.$$

Finally, recalling that in the above we conditioned on the server being busy, we get that $\mathsf{E}[B_r] = \rho\,\mathsf{E}[R]$, thereby yielding (1.74).

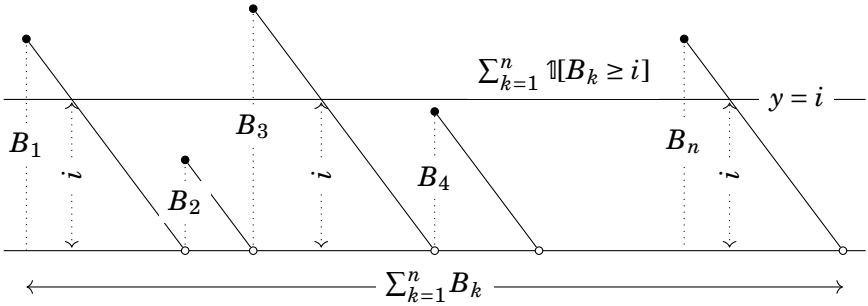


Figure 1.19: The remaining job size as a function of time. The total number of serv periods, which is equal to the number of items arrived, is $\sum_{k=1}^{n} B_k$. A batch crosses t line $y = i$ iff it contains at least i items. Thus, during the service of a batch with i more items, there is precisely one period during which the i-th item of a batch is waiti in queue. Consequently, $\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]$ is the number of periods in which there are p cisely i items waiting at the server. The fraction of periods there are i items is theref $\sum_{k}^{n} \mathbb{1}[B_k \geq i] / \sum_{k}^{n} B_k$.

1.15.10. Show that $2\sum_{i=i}^{\infty} iG(i-1) = \mathsf{E}\left[B^2\right] + \mathsf{E}[B]$.

Alternative derivation Here is an alternative derivation of the fraction of jobs that see i items at the server. Perhaps this is easier. If so, let me know. This is not obligatory reading, BTW.

Let $L_s(s)$ be the number of items (of the batch in service) at the server. Then,

$$Y_S(i,t) = \int_0^t \mathbb{1}[L_S(s) = i]\,\mathrm{d}s$$

is the total time up to time t that there are i items at the server.

1.15.11. Let $\tilde{A}_k$ be the moment the kth batch moves to the server and D_k its departure time. Show that, when $S_{k,i}$ is the service time of the ith item of batch k,

$$\int_{\tilde{A}_k}^{D_k} \mathbb{1}[L_S(s) = i]\,\mathrm{d}s = S_{k,i}\,\mathbb{1}[B_k \geq i],$$

1.15.12. Use the previous exercise to show that

$$Y_S(i,D_n) = \sum_{k=1}^{n} \mathbb{1}[B_k \geq i] S_{k,i}.$$

In Exercise 1.15.12 we will use the result that

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{n} \mathbb{1}[B_k \geq i] S_{k,i}}{\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]} = \mathsf{E}[S \,|\, B \geq i] = \mathsf{E}[S]$$

where the last equality follows from independence. The next exercise is meant to help understand this.

1.15.13. We are given two i.i.d. sequences $A_1, A_2, \ldots$ and $H_1, H_2, \ldots$, where A_k is the kth person's age and H_k the height. Show that

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{n} \mathbb{1}[A_k \leq 15, H_k \leq 1.8]}{\sum_{k=1}^{n} \mathbb{1}[A_k \leq 15]} = \mathsf{P}(H \leq 1.8 \,|\, A \leq 15),$$

where H and A are the limiting r.v.s associated with $\{H_k\}$ and $\{A_k\}$, respectively. With this, show that

$$\lim_{n\to\infty} \frac{\sum_{k=1}^{n} H_k \mathbb{1}[A_k \leq 15]}{\sum_{k=1}^{n} \mathbb{1}[A_k \leq 15]} = \mathsf{E}[H \,|\, A \leq 15].$$

1.15.14. By taking the time average, show that

$$\mathsf{P}(Y_S = i) = \lim_{n} \frac{Y_S(i,D_n)}{D_n} = \rho \frac{G(i-1)}{\mathsf{E}[B]}.$$

1.15.15. Show that $\mathsf{P}(B_r = i) = \mathsf{P}(Y_S = i)$, i.e., the probability that a batch arriving at the system sees $B_r = i$ items at the server is equal to the fraction of time that $Y_s = i$.

With the above exercises we conclude that

$$\mathsf{E}[B_r] = \sum_{i=0}^{\infty} i \mathsf{P}(B_r = i) = \frac{\rho}{\mathsf{E}[B]} \sum_{i=1}^{\infty} iG(i-1).$$

Hints

h.1.15.4 What is the distribution of the batch size B for the $M/M/1$ queue?

h.1.15.5 Use Eq. 1.76. What are $\mathsf{E}\left[B^2\right]$, $\mathsf{E}[B]$ and $\mathsf{V}[B]$ for this case?

h.1.15.6 $f_k = q^{k-1}p$ with $q = 1-p$. Use generating functions to compute $\mathsf{E}[B]$ and $\mathsf{E}\left[B^2\right]$

h.1.15.9 Realize that the inventory process $I(t)$ behaves as $I(t) = S - L(t)$ where $L(t)$ is a suitable queueing process. Refer to Ex. 1.10.12 for further background.

h.1.15.10 Use Exercises 1.2.21 and 1.2.22.

h.1.15.12 Observe that n batches have been served at time D_n.

h.1.15.13 Divide and multiply by $1/n$.

h.1.15.14 Take the middle term and divide and multiply by suitable other counting functions along the sample path, such as $\sum_{k=1}^{n} \mathbb{1}[B_k \geq i]$ and D_n. Also use rate stability, i.e., $\lambda = \delta$.

Solutions

s.1.15.1 For stability, it is necessary that the service rate $\mu = 1/\mathsf{E}[S]$ for items is larger than the rate at which items arrive. Hence,

$$\rho = \lambda\,\mathsf{E}[B]\,\mathsf{E}[S] = \frac{\lambda\,\mathsf{E}[B]}{\mu} < 1.$$

s.1.15.2

$$\mathsf{E}\left[W_{Q,b}\right] = \mathsf{E}[L]\,\mathsf{E}[S] = (\mathsf{E}\left[L_{Q,b}\right]\,\mathsf{E}[B] + \mathsf{E}[B_r])\mathsf{E}[S].$$

With Little's law, $\mathsf{E}\left[L_{Q,b}\right] = \lambda\,\mathsf{E}\left[W_{Q,b}\right]$,

$$\mathsf{E}\left[W_{Q,b}\right] = \lambda\,\mathsf{E}[S]\,\mathsf{E}[B]\,\mathsf{E}\left[W_{Q,b}\right] + \mathsf{E}[S]\,\mathsf{E}[B_r] = \rho\,\mathsf{E}\left[W_{Q,b}\right] + \mathsf{E}[S]\,\mathsf{E}[B_r],$$

hence,

$$\mathsf{E}\left[W_{Q,b}\right] = \frac{\mathsf{E}[B_r]}{1-\rho}\,\mathsf{E}[S].$$

s.1.15.3 We have

$$\frac{\mathsf{E}\left[B^2\right]}{\mathsf{E}[B]} = \frac{\mathsf{E}\left[B^2\right]}{(\mathsf{E}[B])^2}\,\mathsf{E}[B] = \frac{(\mathsf{E}[B])^2 + \mathsf{V}[B]}{(\mathsf{E}[B])^2}\,\mathsf{E}[B] = (1 + C_s^2)\,\mathsf{E}[B].$$

s.1.15.4 For the $M/M/1$ queue, each job contains just one item. Thus, $B \equiv 1$, hence $\mathsf{P}(B=1)=1$, $\mathsf{E}\left[B^2\right]=\mathsf{E}[B]=1$. Therefore, $\mathsf{E}[B_r(M/M/1)]=\rho$, and $\mathsf{E}[L(M/M/1)]=\rho/(1-\rho)$.

s.1.15.5 As B is constant and equal to 3, $\mathsf{E}\left[B^2\right]=9$, hence $\mathsf{V}[B]=0$, hence $C_s^2=0$. Also, $\rho=\lambda\,\mathsf{E}[B]/\mu=1\cdot 3/6=1/2$. Hence

$$\mathsf{E}[L]=\frac{1}{2}\frac{1/2}{1-1/2}\cdot 3+\frac{1}{2}\frac{1/2}{1-1/2}.$$

s.1.15.6 We need $\mathsf{V}[B]$ and $\mathsf{E}[B]$. Consider

$$\begin{aligned}\phi(z)=\mathsf{E}\left[z^B\right]&=\sum_{k=0}^{\infty}z^k\mathsf{P}(B=k)=\sum_{k=0}^{\infty}z^k f_k\\&=\sum_{k=0}^{\infty}z^k pq^{k-1}=\frac{p}{q}\sum_{k=0}^{\infty}(qz)^k=\frac{p}{q}\frac{1}{1-qz}.\end{aligned}$$

Then

$$\mathsf{E}[B]=\phi'(1)=\frac{p}{q}\frac{q}{(1-qz)^2}\bigg|_{z=1}=\frac{p}{(1-q)^2}=\frac{1}{p},$$

and

$$\mathsf{E}[B(B-1)]=\phi''(1)=\frac{p}{q}\frac{2q^2}{(1-qz)^3}\bigg|_{z=1}=2\frac{q}{p^2}=\frac{2}{p^2}-\frac{2}{p}.$$

Hence,

$$\mathsf{E}\left[B^2\right]=\frac{2}{p^2}-\frac{2}{p}+\mathsf{E}[B]=\frac{2}{p^2}-\frac{1}{p},$$

and

$$\mathsf{V}[B]=\mathsf{E}\left[B^2\right]-(\mathsf{E}[B])^2=\frac{2}{p^2}-\frac{1}{p}-\frac{1}{p^2}=\frac{1}{p^2}-\frac{1}{p},$$

and

$$C_s^2=\frac{\mathsf{V}[B]}{(\mathsf{E}[B])^2}=p^2\left(\frac{1}{p^2}-\frac{1}{p}\right)=1-p$$

Then,

$$(1+C_s^2)/2=1-p/2.$$

and

$$\mathsf{E}[L]=\left(1-\frac{p}{2}\right)\frac{\rho}{1-\rho}\frac{1}{p}+\frac{1}{2}\frac{\rho}{1-\rho}=\frac{\rho}{1-\rho}\frac{1}{p}.$$

Can we check this in a simple way? If $\mathsf{P}(B=1)=f_1=p=1$, then $\mathsf{E}[L]=\rho/(1-\rho)$. Thus, we get the result for the $M/M/1$ queue. The result is at least consistent with earlier work.

s.1.15.7 Start with the simple case, $B \equiv 2$. Then $\mathsf{V}[B] = 0$ and $\mathsf{E}[B] = 2$. The load is $\rho = \lambda \mathsf{E}[B]\,\mathsf{E}[S] = 1 \cdot 2 \cdot 25/60 = 5/6$. hence,

$$\mathsf{E}[L] = \frac{1}{2}\frac{5/6}{1/6}2 + \frac{1}{2}\frac{5/6}{1/6} = 5 + \frac{5}{2}.$$

Now the other case. $\mathsf{E}\left[B^2\right] = (1+4+9)/3 = 14/3$. Hence, $\mathsf{V}[B] = 14/3 - 4 = 2/3$. Hence,

$$C_s^2 = \frac{\mathsf{V}[B]}{(\mathsf{E}[B])^2} = \frac{2/3}{4} = \frac{1}{6}.$$

And thus,

$$\mathsf{E}[L] = \frac{1+1/6}{2}\frac{5/6}{1/6}2 + \frac{1}{2}\frac{5/6}{1/6} = \frac{7}{6}5 + \frac{5}{2}.$$

If we divide these two answers, we see that the ratio between $\mathsf{E}[L]$ for both answers is 10/9. In other words, we can reduce about 10% of the number of items in the system by working in fixed batch sizes.

s.1.15.8

$$\frac{\mathsf{E}\left[L(M^X/M/1)\right]}{\mathsf{E}[L(M/M/1)]} = \frac{\mathsf{E}[B_r]}{\rho} = \frac{\mathsf{E}\left[B^2\right]}{2\mathsf{E}[B]} + \frac{1}{2}.$$

With this we can check whether this condition

$$1 \le \frac{\mathsf{E}\left[L(M^X/M/1)\right]}{\mathsf{E}[L(M/M/1)]} = \frac{\mathsf{E}\left[B^2\right]}{2\mathsf{E}[B]} + \frac{1}{2}$$

is always true. Clearly, it reduces to

$$\mathsf{E}[B] \le \mathsf{E}\left[B^2\right].$$

Multiply this by $\mathsf{E}[B]$ for reasons to become clear presently to get

$$(\mathsf{E}[B])^2 \le \mathsf{E}\left[B^2\right]\mathsf{E}[B].$$

So, the initial inequality is converted to this, and we like to know whether this always true.

To see this, we can use Jensen's inequality $\phi(\mathsf{E}[X]) \le \mathsf{E}\left[\phi(X)\right]$ when ϕ is convex. In this case take $\phi(x) = x^2$, so that Jensen's inequality states that $(\mathsf{E}[B])^2 \le \mathsf{E}\left[B^2\right]$. (BTW, note that Jensen's inequality implies that $\mathsf{V}[X] = \mathsf{E}\left[X^2\right] - (\mathsf{E}[X])^2 \ge 0$.) Now noting that $B \ge 1$, as a job minimally contains one item, we get

$$\begin{aligned}(\mathsf{E}[B])^2 &\le \mathsf{E}\left[B^2\right], \quad \text{by Jensen's inequality} \\ &\le \mathsf{E}\left[B^2\right]\mathsf{E}[B], \quad \text{as } B \ge 1.\end{aligned}$$

Clearly, this is the inequality we tried to show. As a result,

$$1 \leq \frac{\mathsf{E}\left[L(M^X/M/1)\right]}{\mathsf{E}[L(M/M/1)]}$$

for all B.

In conclusion, if work arrives in batches, the average number of jobs in the system, increases, hence the average waiting time increases.

s.1.15.9 Consider a queueing system with job arrival rate $\lambda = 6$ per hour and the jobs have batch sizes as indicated in the problem. The average number of items in the system follows like this

$$\begin{aligned}
\mathsf{E}[B] &= \frac{1+2+3+4}{4} = \frac{5}{2} \\
\mathsf{E}\left[B^2\right] &= \frac{1+4+9+16}{4} = \frac{30}{4} \\
\mathsf{V}\left[B^2\right] &= \frac{30}{4} - \frac{25}{4} = \frac{5}{4} \\
C_s^2 &= \frac{5/4}{25/4} = \frac{1}{5} \\
\rho &= \lambda\,\mathsf{E}[B]\,\mathsf{E}[S] = 6\frac{5}{2}\frac{1}{20} = \frac{3}{4}.
\end{aligned}$$

Hence,

$$\mathsf{E}[L] = \frac{1+1/5}{2}\frac{3/4}{1/4}\frac{5}{2} + \frac{1}{2}\frac{3/4}{1/4} = 6.$$

Thus, if the level is set to $S = 4$, then on average there will be two items short. Clearly, then, S should be at least 6. However, $\mathsf{E}[L]$ is just the average. Roughly speaking, in this case half of the demand will then be lost. So, if we take variability into account, S should be quite a bit bigger than 6.

A more detailed analysis is necessary to determine the right value of S such that not more than a certain fraction of demand is lost. We will address this issue in Section 1.17.

s.1.15.10

$$\begin{aligned}
2\sum_{i=1}^{\infty} iG(i-1) &= 2\sum_{i=0}^{\infty}(i+1)G(i) = 2\sum_{i=0}^{\infty} iG(i) + 2\sum_{i=0}^{\infty} G(i) \\
&= \mathsf{E}\left[B^2\right] - \mathsf{E}[B] + 2\,\mathsf{E}[B].
\end{aligned}$$

s.1.15.11 Only if $B_k \geq i$ there can be an ith item of the batch, and the time this ith item spends at the server is its service time $S_{k,i}$.

s.1.15.12 At the departure time D_n of the nth batch, precisely n batches have been served. Thus, each batch with more than i items, contributed to the integral $\int_0^{D_n} \mathbb{1}[L_S(s)$ with the service time $S_{k,i}$.

s.1.15.13

$$\frac{\sum_{k=1}^n \mathbb{1}[A_k \le 15, H_k \le 1.8]}{\sum_{k=1}^n \mathbb{1}[A_k \le 15]} = \frac{n^{-1}\sum_{k=1}^n \mathbb{1}[A_k \le 15, H_k \le 1.8]}{n^{-1}\sum_{k=1}^n \mathbb{1}[A_k \le 15]} \to \frac{\mathsf{P}(H \le 1.8, A \le 15)}{\mathsf{P}(A \le 15)} = \mathsf{P}(H \le 1.8 \,|\, A \le 15).$$

The next equation is more subtle. Take some $\delta > 0$. It is simple to see that

$$\delta \sum_{m=0}^{\infty} m\, \mathbb{1}[H_k \in [m\delta, (m+1)\delta)] \le H_k \le \delta \sum_{m=0}^{\infty} (m+1)\, \mathbb{1}[H_k \in [m\delta, (m+1)\delta)].$$

Morever, if $\delta \to 0$, the left and right hand side converge. Let us use the left hand side as an approximation for H_k.

$$\begin{aligned}
\frac{\sum_{k=1}^n H_k \mathbb{1}[A_k \le 15]}{\sum_{k=1}^n \mathbb{1}[A_k \le 15]} &\approx \delta \frac{\sum_{k=1}^n \sum_{m=0}^{\infty} m\, \mathbb{1}[A_k \le 15, H_k \in [m\delta, (m+1)\delta)]}{\sum_{k=1}^n \mathbb{1}[A_k \le 15]} \\
&= \delta \sum_{m=0}^{\infty} m \frac{\sum_{k=1}^n \mathbb{1}[A_k \le 15, H_k \in [m\delta, (m+1)\delta)]}{\sum_{k=1}^n \mathbb{1}[A_k \le 15]} \\
&\xrightarrow{n\to\infty} \delta \sum_{m=0}^{\infty} m \frac{\mathsf{P}(A \le 15, H \in [m\delta, (m+1)\delta))}{\mathsf{P}(A \le 15)} \\
&= \delta \sum_{m=0}^{\infty} m \mathsf{P}(H \in [m\delta, (m+1)\delta) \,|\, A \le 15) \\
&\xrightarrow{\delta\to 0} \mathsf{E}[H \,|\, A \le 15].
\end{aligned}$$

Finally, if H and A would be independent (which they are not), then it is easy in the above to take out the condition on $A \le 15$ and conclude that $\mathsf{E}[H \,|\, A \le 15] = \mathsf{E}[H]$.

s.1.15.14 By taking the time average and assuming that all limits exists, we get

$$\begin{aligned}
\mathsf{P}(Y_S = i) &= \lim_n \frac{Y_S(i, D_n)}{D_n} \\
&= \lim_n \frac{n}{D_n} \cdot \frac{\sum_{k=1}^n \mathbb{1}[B_k \ge i]}{n} \cdot \frac{\sum_{k=1}^n \mathbb{1}[B_k \ge i] S_{k,i}}{\sum_{k=1}^n \mathbb{1}[B_k \ge i]}.
\end{aligned}$$

Observe now that the first fraction becomes the departure rate δ, the second is the fraction of batches containing at least i items, and the third converges to $\mathsf{E}[S]$ as the service times of the items are i.i.d. Thus, by rate stability and using that $G(i) = \mathsf{P}(B > i)$,

$$\mathsf{P}(Y_S = i) = \delta \mathsf{P}(B \geq i)\,\mathsf{E}[S] = \lambda G(i-1)\mathsf{E}[S] = \rho \frac{G(i-1)}{\mathsf{E}[B]},$$

where in the last step we use $\rho = \lambda\,\mathsf{E}[S]\,\mathsf{E}[B]$.

s.1.15.15 Follows right away from the PASTA property.

1.16 M/G/1 Queue: Expected Waiting Time

Theory and Exercises

Let's focus on one queue in a supermarket, served by one cashier, and assume that customers do not jockey, i.e., change queue. What can we say about the average waiting time in queue if service times are not exponential, like in the $M/M/1$ queue, but have a more general distribution? One of the celebrated results of queueing theory is the Pollackzek-Khintchine formula by which we can compute the average waiting time formula for the $M/G/1$ queue. In this section we derive this result by means of sample path arguments.

Recall that Eq. (1.68) states that

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

This equation is valid for the $M/G/1$ queue, as in the derivation we used the PASTA property but did not need that the service time distribution is exponential. Thus, we need to compute the average remaining service time $\mathsf{E}[S_r]$ for generally distributed service times. We turn to this task now.

Consider the kth job of some sample path the $M/G/1$ queueing process. This job requires S_k units of service, let its service time start at time $\tilde{A}_k$ so that it departs the server at time $D_k = \tilde{A}_k + S_k$, c.f. Figure 1.20. Define

$$R_k(s) = (D_k - s)\,\mathbb{1}[\tilde{A}_k \leq s < D_k]$$

as the remaining service time at time s of job k. To see this, observe that when the time $s \in [\tilde{A}_k, D_k)$, the remaining service time until job k departs is $D_k - s$, while if $s \notin [\tilde{A}_k, D_k)$, job k is not in service so it cannot have any remaining service as well.

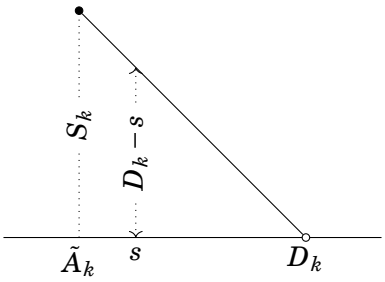


Figure 1.20: Remaining service time.

For the average remaining service time, we can add all remaining service times up to some time t, and then divide by t, specifically,

$$R(t) = \frac{1}{t}\int_0^t \sum_{k=1}^{\infty} R_k(s)\,\mathrm{d}s,$$

so that we can define

$$\mathsf{E}[S_r] = \lim_{t\to\infty} R_t,$$

provided this limit exists along the sample path.

Now observe that as t can be somewhere in a service interval, it is easiest to consider $R(t)$ at departure times $\{D_n\}$, as at a time D_n precisely n jobs departed. Thus,

$$R(D_n) = \frac{1}{D_n}\int_0^{D_n} \sum_{k=1}^{\infty} R_k(s)\,\mathrm{d}s = \frac{1}{D_n}\sum_{k=1}^{n}\int_0^{D_n} R_k(s)\,\mathrm{d}s,$$

the interchange being allowed by the positivity of $R_k(s)$.

1.16.1. Show that for $k < n$, so that $D_k < D_n$, $\int_0^{D_n} R_k(s)\,\mathrm{d}s = \frac{S_k^2}{2}$.

With this exercise

$$R(D_n) = \frac{1}{D_n}\sum_{k=1}^{n}\frac{S_k^2}{2} = \frac{n}{D_n}\frac{1}{n}\sum_{k=1}^{n}\frac{S_k^2}{2}.$$

Finally, assuming that along this sample path, $D_n \to \infty$ as $t\to\infty$, $n/D_n \to \delta$, $\delta = \lambda$ by rate stability, and the limit $n^{-1}\sum_{k=1}^{n} S_k^2$ exists as $n\to\infty$, we get

$$\mathsf{E}[S_r] = \lim_{D_n\to\infty} R(D_n) = \frac{\lambda}{2}\mathsf{E}\left[S^2\right]. \tag{1.77}$$

1.16.2. Show from Eq. (1.77) that

$$\mathsf{E}[S_r \,|\, S_r > 0] = \frac{\mathsf{E}\left[S^2\right]}{2\,\mathsf{E}[S]}. \tag{1.78}$$

1.16.3. Show that $\mathsf{E}[S_r \,|\, S_r > 0] = (2\mu)^{-1}$ when $\mathsf{E}[S] = \mu^{-1}$ and $\mathsf{V}[S] = 0$.

1.16.4. Show that $\mathsf{E}[S_r \,|\, S_r > 0] = 2/(3\mu)$ when $S \sim U[0, 2/\mu]$.

1.16.5. Show that $\mathsf{E}[S_r \,|\, S_r > 0] = \mu^{-1}$ when $S \sim \exp(\mu)$.

Thus,

$$\mathsf{E}\left[W_Q\right] = \frac{1}{1-\rho}\,\mathsf{E}[S_r] = \frac{1}{2}\,\frac{\lambda\,\mathsf{E}\left[S^2\right]}{1-\rho}.$$

1.16.6. Show that

$$\lambda\,\mathsf{E}\left[S^2\right] = (1 + C_s^2)\rho\,\mathsf{E}[S], \quad \text{where } C_s^2 = \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} \tag{1.79}$$

is the *square coefficient of variation*.

With the simplification of the above exercise we have obtained the fundamentally important *Pollackzek-Khintchine* (PK) formula for the average waiting time in queue:

$$\mathsf{E}\left[W_Q\right] = \frac{1 + C_s^2}{2}\,\frac{\rho}{1-\rho}\,\mathsf{E}[S]. \tag{1.80}$$

The problems below will clarify how useful this result is.

1.16.7. 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 $\mathsf{E}\left[L_Q\right]$, $\mathsf{E}[L]$, $\mathsf{E}\left[W_Q\right]$, $\mathsf{E}[W]$.

1.16.8. Consider a workstation with just one machine. Jobs arrive, roughly, in accordance to 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?

1.16.9. (Hall 5.16) The manager of a small firm would like to determine which of two people to hire. One employee is fast, on average, but is 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 minute, with a standard deviation of 0.1 minute. If the arrival rate is Poisson with rate 20 per hour, which employee would minimize $\mathsf{E}\left[L_Q\right]$? Which would minimize $\mathsf{E}[L]$?

1.16.10. Show that when services are exponential, the expected waiting time $\mathsf{E}\left[W_Q(M/G/1)\right]$ reduces to $\mathsf{E}\left[W_Q(M/M/1)\right]$.

1.16.11. Compute $\mathsf{E}\left[W_Q\right]$ and $\mathsf{E}[L]$ for the $M/D/1$ queue. Assume that the service time is always T. Compare $\mathsf{E}[L(M/D/1)]$ to $\mathsf{E}[L(M/M/1)]$ where the mean service time is the same in both cases.

1.16.12. Compute $\mathsf{E}[L]$ for the $M/G/1$ queue with $S \sim U[0,A]$.

1.16.13. Show that for the $M/G/1$ queue, the expected idle time is $\mathsf{E}[I] = 1/\lambda$.

1.16.14. Use Exercise 1.7.10 to show that for the $M/G/1$ queue, the expected busy time is given by $\mathsf{E}[B] = \mathsf{E}[S]/(1-\rho)$.

1.16.15. What is the utilization of the $M/G/1/1$ queue?

1.16.16. For the $M/G/1/1$ queue what is the fraction of jobs rejected (hence, what is the fraction of accepted jobs)?

1.16.17. Why is the fraction of lost jobs at the $M/G/1/1$ queue not necessarily the same as for a $G/G/1/1$ queue with the same load?

Hints

h.1.16.1 Graphically, the result follows from Figure 1.20.

h.1.16.3 $\mathbb{V}[S] = 0$ implies that S is deterministic.

h.1.16.12 Integrate $A^{-1}\int x\,\mathrm{d}x$, and likewise for the second moment.

h.1.16.13 is the average time between two arrivals? Observe that the interarrivals are memoryless, hence the average time until the next arrival after the server becomes idle is also $1/\lambda$.

h.1.16.15 The job in the system is also in service (it is an $M/G/1/1$ queue...). The average idle time is $1/\lambda$. The average busy time is $\mathsf{E}[S]$–as there are no jobs in queue. The load ρ must also be equal to the fraction of time the server is busy.

h.1.16.16 The rate of accepted jobs is $\lambda\pi(0)$. What is the load of these jobs? Equate this to $1-\pi(0$ as this must also be the load. Then solve for $\pi(0)$.

h.1.16.17 Provide an example.

Solutions

s.1.16.1 If $k < n$, $D_k < D_n$, and then

$$\begin{aligned}\int_0^{D_n} R_k(s)\,\mathrm{d}s &= \int_0^{D_n} (D_k - s)\,\mathbb{1}[\tilde{A}_k \le s < D_k]\,\mathrm{d}s = \int_{\tilde{A}_k}^{D_k} (D_k - s)\,\mathrm{d}s \\ &= \int_{\tilde{A}_k}^{\tilde{A}_k+S_k} (\tilde{A}_k + S_k - s)\,\mathrm{d}s = \int_0^{S_k} (S_k - s)\,\mathrm{d}s \\ &= \int_0^{S_k} s\,\mathrm{d}s = \frac{S_k^2}{2}.\end{aligned}$$

s.1.16.2 From Eq. (1.66) and the sentences above this equation, we have that

$$\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \,|\, S_r > 0] = \lambda\,\mathsf{E}[S]\,\mathsf{E}[S_r \,|\, S_r > 0].$$

Hence,

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

s.1.16.3 When S is deterministic, $\mathsf{P}\left(S = 1/\mu\right) = 1$. Therefore, $\mathsf{E}[S] = \mu^{-1}$ and $\mathsf{E}\left[S^2\right] = \mu^{-2}$. Now use (1.78).

s.1.16.4 When S is uniform on $[0, \alpha]$,

$$\mathsf{E}[S] = \alpha^{-1}\int_0^\alpha x\,\mathrm{d}x = \frac{\alpha^2}{2\alpha} = \frac{\alpha}{2}, \qquad \mathsf{E}\left[S^2\right] = \alpha^{-1}\int_0^\alpha x^2\,\mathrm{d}x = \frac{\alpha^2}{3}.$$

Thus, with (1.78), $\mathsf{E}[S_r \,|\, S_r > 0] = (\alpha^2/3)/(2\alpha/2) = \alpha/3$. Finally, take $\alpha = 2/\mu$.

s.1.16.5

$$\begin{aligned}\mathsf{E}[S] &= \mu\int_0^\infty xe^{-\mu x}\,\mathrm{d}x = 1/\mu,\\ \mathsf{E}\left[S^2\right] &= \mu\int_0^\infty x^2e^{-\mu x}\,\mathrm{d}x = x^2e^{-\mu x}\Big|_0^\infty + 2\int_0^\infty xe^{-\mu x}\,\mathrm{d}x\\ &= 2\frac{x}{\mu}e^{-\mu x}\Big|_0^\infty + \frac{2}{\mu}\int_0^\infty e^{-\mu x}\,\mathrm{d}x = \frac{2}{\mu^2}.\end{aligned}$$

s.1.16.6 Use that

$$\frac{\mathsf{E}\left[S^2\right]}{(\mathsf{E}[S])^2} = \frac{(\mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2) + (\mathsf{E}[S])^2}{(\mathsf{E}[S])^2} = \frac{\mathsf{V}[S] + (\mathsf{E}[S])^2}{(\mathsf{E}[S])^2} = C_s^2 + 1.$$

Then

$$\lambda\,\mathsf{E}\left[S^2\right] = \frac{\mathsf{E}\left[S^2\right]}{(\mathsf{E}[S])^2}\lambda(\mathsf{E}[S])^2 = \frac{\mathsf{E}\left[S^2\right]}{(\mathsf{E}[S])^2}\rho\,\mathsf{E}[S] = (1 + C_s^2)\rho\,\mathsf{E}[S].$$

s.1.16.7 First the load.

```
>>> labda = 5./60 # arrivals per minute
>>> a = 4.
>>> b = 6.
>>> ES = (a+b)/2.  # service time in minutes
>>> rho = labda*ES
>>> rho
0.41666666666666663
```

Next, the variance and SCV. With this the waiting times follow right away.

```
>>> Var = (b-a)*(b-a)/12.
>>> SCV = Var/(ES**2)


>>> Wq = (1+SCV)/2.*rho/(1.-rho)*ES
>>> Wq # in minutes
1.8095238095238095
>>> Wq/60. # in hour
0.03015873015873016


>>> W = Wq + ES
>>> W
6.809523809523809
>>> Lq = labda*Wq
>>> Lq
0.15079365079365079
>>> L = labda*W
>>> L
0.5674603174603174
```

s.1.16.8 $\mathsf{E}[W_Q] = 4.5$ h. $\rho = \lambda \mathsf{E}[S] = (3/8) \cdot 2 = 3/4$.

One way to increase the capacity/reduce the average service time is to choose $\mathsf{E}[S] = 1$ hour and reduce C_s^2 to 1/4. Of course, there are many more ways. Reducing C_S^2 to zero is (nearly) impossible or very costly. Hence, ρ must go down, i.e., capacity up or service time down. Another possibility is to plan the arrival of jobs, i.e., reduce the variability in the arrival process. However, typically this is not possible. For instance, would you accept this as a customer?

s.1.16.9 The arrival process is assumed to be Poisson. There is also just one server. Hence, we can use the PK formula to compute the average queue length.

```
>>> labda = 20./60 # per hour
>>> ES = 2. # hour
>>> sigma = 1.
>>> SCV = sigma*sigma/(ES*ES)
>>> rho = labda*ES
>>> rho
```

```
0.6666666666666666

>>> Wq = (1+SCV)/2 * rho/(1-rho) * ES
>>> Wq
2.4999999999999996
>>> Lq = labda * Wq # Little's law
>>> Lq
0.8333333333333331
>>> W = Wq + ES
>>> W
4.5
>>> L = labda * W
>>> L
1.5

>>> ES = 2.1
>>> SD = 0.1
>>> SCV = SD**2/ES**2
>>> rho = labda*ES
>>> rho
0.7

>>> Lq = rho**2/(1.-rho)*(1.+SCV)/2.
>>> Lq
0.8185185185185182
>>> L = rho + Lq
>>> L
1.5185185185185182
```

s.1.16.10 Since the SCV for the exponential distribution is 1, hence $C_s^2 = 1$, we get

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

which we also found in a previous section.

s.1.16.11 For the $M/D/1$ the service time is deterministic. Thus, the service time $S = T$ always; recall that S is deterministic under the assumptions in the exercise. Therefore, we must have that $\mathsf{P}(S \leq T) = 1$. As an immediate consequence, $\mathsf{P}(S > T) = 1 - \mathsf{P}(S \leq T) = 0$. Also, $\mathsf{P}(S \leq x) = 0$ for all $x < T$. Thus, all probability

mass is concentrated on the point T, thus, it is also impossible that the service time is smaller than T. (If you are into maths, then you should be aware of the fact that 'impossible' and 'almost surely' are not the same. Thus, my using of the word 'impossible' is not completely correct.)

To compute the SCV, I first compute $\mathsf{E}[S]$, then $\mathsf{E}\left[S^2\right]$ and then $\mathsf{V}[S] = \mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2$, and use the definition of coefficient of variation. I use these steps time and again.

First, $\mathsf{E}[S] = T$ since $\mathsf{P}(S = T) = 1$. We can also obtain this in a more formal way. The distribution $F(x)$ of S has a, so-called, atom at $x = T$, such $F(T) - F(T-) = 1$. Using this,

$$\mathsf{E}[S] = \int_0^\infty x \, \mathrm{d}F(x) = T \cdot (F(T) - F(T-)) = T.$$

Similarly

$$\mathsf{E}\left[S^2\right] = \int_0^\infty x^2 \, \mathrm{d}F(x) = T^2 \cdot (F(T) - F(T-)) = T^2.$$

Hence $\mathsf{V}[S] = \mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2 = 0$, hence $C_s^2 = 0$.

From the Pollackzek-Khintchine formula, the first term is $1 + C_s^2 = 1$ for the $M/D/1$ queue, and $1 + C_s^2 = 2$ for the $M/M/1$ queue. Hence,

$$\mathsf{E}\left[W_Q(M/D/1)\right] = \frac{\mathsf{E}\left[W_Q(M/M/1)\right]}{2}$$

thus,

$$\mathsf{E}\left[L_Q(M/D/1)\right] = \frac{\mathsf{E}\left[L_Q(M/M/1)\right]}{2}.$$

In both cases the expected service times are equal to T so that the loads are the same. Thus,

$$\begin{aligned}\mathsf{E}[L(M/D/1] &= \mathsf{E}\left[L_Q(M/D/1)\right] + \rho = \frac{\mathsf{E}\left[L_Q(M/M/1)\right]}{2} + \rho \\ &= \frac{\rho^2}{2(1-\rho)} + \rho = \frac{\rho(2-\rho)}{2(1-\rho)}.\end{aligned}$$

s.1.16.12

$$\begin{aligned}\mathsf{E}[S] &= A/2, \\ \mathsf{E}\left[S^2\right] &= \int_0^A x^2 \, \mathrm{d}x/A = A^2/3 \\ \mathsf{V}[S] &= A^2/3 - A^2/4 = A^2/12\end{aligned}$$

$$
\begin{aligned}
C_s^2 &= (A^2/12)/(A^2/4) = 1/3,\\
\rho &= \lambda A/2,\\
\mathsf{E}\left[W_Q\right] &= \frac{1+C_s^2}{2}\frac{\lambda A/2}{1-\lambda A/2}\frac{A}{2},\\
\mathsf{E}[W] &= \mathsf{E}\left[W_Q\right] + \frac{A}{2}\\
\mathsf{E}[L] &= \lambda\,\mathsf{E}[W].
\end{aligned}
$$

s.1.16.13 Since the interarrival times are memoryless, the expected time to the first arrival after a busy time stops is also $\mathsf{E}[X] = 1/\lambda$.

s.1.16.14 Since $\rho = \lambda\,\mathsf{E}[S]$, $\mathsf{E}[I] = 1/\lambda$ and from Exercise 1.7.10

$$\mathsf{E}[B] = \frac{\rho}{1-\rho}\,\mathsf{E}[I] = \frac{\lambda\,\mathsf{E}[S]}{1-\rho}\frac{1}{\lambda}.$$

The result follows.

s.1.16.15 The system can contain at most 1 job. Necessarily, if the system contains a job, this job must be in service. All jobs that arrive while the server is busy are blocked, in other words, rejected. Just after a departure, the average time until the next arrival is $1/\lambda$, then a service starts with an average duration of $\mathsf{E}[S]$. After this departure, a new cycle start. Thus, the utilization is $\mathsf{E}[S]/(1/\lambda + \mathsf{E}[S]) = \lambda\,\mathsf{E}[S]/(1+\lambda\,\mathsf{E}[S])$. Since not all jobs are accepted, the utilization ρ cannot be equal to $\lambda\,\mathsf{E}[S]$.

s.1.16.16 Let $p(0)$ be the fraction of time the server is idle. By PASTA, $\pi(0) = p(0)$. Thus, the rate of accepted jobs is $\lambda\pi(0)$. Therefore, the departure rate $\delta = \lambda\pi(0)$. The loss rate is $\lambda - \delta = \lambda(1-\pi(0))$.

Since $\lambda\pi(0)$ is the rate at which jobs enter the system, the load must be $\lambda\pi(0)\mathsf{E}[S]$. Since the load is also $1-\pi(0)$, it follows from equating that

$$\lambda\pi(0)\mathsf{E}[S] = 1-\pi(0) \iff \pi(0) = \frac{1}{1+\lambda\,\mathsf{E}[S]} \iff 1-\pi(0) = \frac{\lambda\,\mathsf{E}[S]}{1+\lambda\,\mathsf{E}[S]},$$

which is the same an in the previous problem.

Note that $\delta < \lambda$, as it should be the case.

s.1.16.17 Typically, in the $G/G/1$ queue, the arrivals do not see time-averages. Consequently, the fraction of arrivals that are blocked is not necessarily equal to the utilization ρ.

Again, take jobs with duration 59 minutes and interarrival times of 1 hour. The load is 59/60, but no job is lost, also not in the $G/G/1/1$ case. Thus, $\delta = \lambda$ in this case.

1.17 $M^X/M/1$ Queue Length Distribution

Theory and Exercises

In Sections 1.15 and 1.16 we established the Pollackzek-Khintchine formula for the waiting times of the $M^X/M/1$ queue and $M/G/1$ queue, respectively. To compute more difficult performance measures, for instance the loss probability $\mathsf{P}(L > n)$, we need expressions for the stationary distribution of the number of jobs in the system $p(n) = \mathsf{P}(L = n)$. Here we present a numerical, recursive, scheme to compute the stationary distribution of the number of items in an $M^X/M/1$ queue.

To find the distribution $p(n)$, which is equal to $\pi(n)$ by PASTA, we turn to level-crossing arguments. However, the arguments that lead to the level-crossing equation (1.42) need to generalized. To see this, we consider an example. If $L(t) = 3$, the system contains 3 items (but not necessarily 3 jobs.) 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 $B_k = 20$, then right after the arrival of job k the system contains 23 items, that is, $L(A_k) = 3 + 20 = 23$. In this case, at the arrival of job k, all levels between states 3 and 23 are crossed.

The left panel in Figure 1.21 demonstrates the up- and downcrossings in more general terms. Level n can be up-crossed from below from many states, in fact from any level $m, 0 \le m < n$. However, it can only be down-crossed from state $n+1$.

As always with level-crossing arguments, we turn to counting how often level n is upcrossed and downcrossed as a function of time. The downcrossing rate is easy; it is exactly the same as for the $M/M/1$ queue. Thus, from Eq. (1.43b),

$$\frac{D(n,t)}{t} = \frac{D(n,t)}{Y(n+1,t)} \frac{Y(n+1,t)}{t} \tag{1.81}$$

which, as $t \to \infty$, converges to the downcrossing rate

$$\mu(n+1)p(n+1) = \mu\pi(n+1), \tag{1.82}$$

where we use that $\mu(n+1) = \mu$ by the memoryless property of the service times and PASTA to see that $\pi(n+1) = p(n+1)$. Observe that this part was easy because there is just one way (i.e., there is just one arrow from right to left in Figure 1.21) to downcross level n, namely from $n+1$ to n.

Counting upcrossings requires quite some more work. Observe that $\mathbb{1}[L(A_k-) \le n] = 1$ only when the kth job sees n or less items in the system, and $\mathbb{1}[L(A_k) > n] = 1$ only when after the kth arrival the system contains more than n items. Thus,

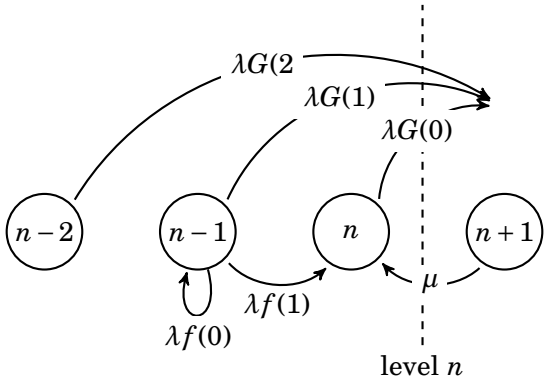


Figure 1.21: Level crossing of level n. Observe that when the system is in state $n-2$, t arrival of any batch larger than 2 ensures that level n is crossed from below. The rate which such events happen is $\lambda G(2)$. Similarly, in state $n-1$ the arrival of any batch larg than one item ensures that level n is crossed, and this occurs with rate $\lambda G(1)$, and so o

$\mathbb{1}[L(A_k-) \le n]\,\mathbb{1}[L(A_k) > n] = 1$ only when the kth arrival generates an upcrossing of level n.

From Figure 1.21 we see that an upcrossing can be decomposed into:

$$\begin{aligned}
&\mathbb{1}[L(A_k-) \le n]\,\mathbb{1}[L(A_k) > n] \\
&= \mathbb{1}[L(A_k-) = n]\,\mathbb{1}[B_k > 0] + \mathbb{1}[L(A_k-) = n-1]\,\mathbb{1}[B_k > 1] + \cdots + \mathbb{1}[L(A_k-) = 0]\,\mathbb{1}[B_k > n] \\
&= \sum_{m=0}^{n} \mathbb{1}[L(A_k-) = m]\,\mathbb{1}[B_k > n-m].
\end{aligned}$$

In other words, paths that upcross level n require that any job that sees m ($m \le n$) in the system upon arrival must bring a batch larger than $n-m$ items.

Next, 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 $n-m$.

1.17.1. Show that $A(n,n,t) = A(n,t)$, where $A(n,t)$ is defined by Eq. 1.40a.

As in Section 1.9, we are primarily intested in long-run averages. For this purpose, observe that we can write

$$\frac{A(m,n,t)}{t} = \frac{A(t)}{t}\,\frac{A(m,t)}{A(t)}\,\frac{A(m,n,t)}{A(m,t)}. \tag{1.83}$$

By the assumptions of Section 1.12, $A(t)/t \to \lambda$ and $A(m,t)/A(t) \to \pi(m)$. Now, provided the limit exists, we define

$$\lim_{t\to\infty} \frac{A(m,n,t)}{A(m,t)} = \lim_{t\to\infty} \frac{\sum_{k=1}^{A(t)} \mathbb{1}[L(A_k-)=m, B_k > n-m]}{\sum_{k=1}^{A(t)} \mathbb{1}[L(A_k-)=m]} = \mathsf{P}(B > n-m \,|\, L(A-)=m), \tag{1.84}$$

where the random variable $L(A-)$ denotes the number in the system seen by an arbitrary arrival.

1.17.2. Show that $\mathsf{P}(B > n-m \,|\, L(A-)=m) = \mathsf{P}(B > n-m)$

By the above exercise,

$$\lim_{t\to\infty} \frac{A(m,n,t)}{A(n,t)} = \mathsf{P}(B > n-m) = G(n-m).$$

By combining the above and making the usual assumptions about the existence of all limits involved we find

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

1.17.3. Provide an interpretation of the above result in terms of a thinned Poisson arrival process.

The last step is to relate the up- and downcrossing rates. Clearly, $\sum_{m=0}^{n} A(m,n,t)$ is the total number of times level n is upcrossed up to time t. By level crossing,

$$\sum_{m=0}^{n} A(m,n,t) \approx D(n,t).$$

Thus, taking the limit $t \to \infty$ in this equation, we conclude that

$$\lambda \sum_{m=0}^{n} G(n-m)\pi(m) = \mu\pi(n+1). \tag{1.85}$$

1.17.4. Show that Eq. (1.85) reduces to $\mu p(n+1) = \lambda p(n)$ for the $M/M/1$ case.

1.17.5. With $\alpha = \lambda/\mu$, show that that *unnormalized* state probabilities are given by

$$\pi(0) = 1 \qquad \pi(1) = \alpha$$
$$\pi(2) = \alpha^2 + \alpha G(1), \qquad \pi(3) = \alpha[\alpha^2 + 2\alpha G(1)) + G(2)].$$

We leave the rest to the computer to continue with this.

It is left to find the normalization constant. As this recursion does not lead to a closed form expression for $p(n)$, such as Eq. (1.49), we need to use a criterion to stop this iterative procedure. Finding general conditions on when to stop is not directly easy, but a pragmatic approach is to simply stop at some (large) number N such that $p(k) \ll p(0)$ for $k > N$ and is steadily decreasing. (An interesting question, why should it decrease monotonically after some, large, N?) Then take $G = \sum_{i=0}^{N} \pi(i)$ as the normalization constant, so that $\pi(0) = 1/G$, $\pi(1) = \alpha/G$, and so on. While this is a practical approach, getting formal bounds on a proper size of N requires more work than we can do here.

Now that we have the state probabilities, we can compute the *excess probability*

$$\mathsf{P}(L > n) = \sum_{i=n+1}^{\infty} \pi(i). \tag{1.86}$$

This performance measure is of interest in inventory systems or—recall the supermarket example discussed in the Introduction—when there are costs associated with the occurence of long queues.

An interesting extension is to consider a queueing process with batch services, i.e., the $M/M^Y/1$ queue. Construction a recursion for the steady-state probabilities $p(n)$ for this case is not hard, in fact, mostly analogous to Eq. (1.85). However, solving the recursion appears to be quite a bit harder; we will not discuss this further here

1.17.6. Why is Eq. (1.64), i.e., $\pi(n) = \delta(n)$, not true for the $M^X/M/1$ batch queue?

Let us use recursion Eq. (1.85) for $\pi(n)$ to derive an expression for the expected number of units of work $\mathsf{E}[L]$ in the system.

1.17.7. Show that

$$\mu \mathsf{E}[L] = \mu \sum_{n=0}^{\infty} n\pi(n) = \lambda \frac{\mathsf{E}\left[B^2\right]}{2} + \lambda \mathsf{E}[B]\,\mathsf{E}[L] + \lambda \frac{\mathsf{E}[B]}{2}, \tag{1.87}$$

With this we can check a result of Section 1.15.

1.17.8. Use Eq. (1.87) and the definition for $\rho = \lambda \mathsf{E}[B]/\mu$ to show that

$$(1-\rho)\mathsf{E}[L] = \frac{\lambda}{\mu}\frac{\mathsf{E}\left[B^2\right]}{2} + \frac{\rho}{2}.$$

1.17.9. Implement the recursion (1.85) in a computer program for the case $f(1) = f(2) = f(3) = 1/3$ (recall $\mathsf{P}(B = k) = f_k$). Take $\lambda = 1$ and $\mu = 3$.

1.17.10. (Finite queueing systems) 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, it is necessary to specify an acceptance, or equivalently a rejection, policy. Three common rules are

1. Complete rejection: if a batch does not fit entirely into the system, it will be rejected completely.
2. Partial acceptance: accept whatever fits of a batch, and reject the rest.
3. Complete acceptance: accept all batches that arrive when the system contains K or less jobs, and reject otherwise.

Derive a set of recursions, analogous to Eq. (1.85), to compute $\pi(n)$ for these three different acceptance rules.

Hints

h.1.17.5 For $n = 1$, we have $\mu\pi(1) = \lambda\pi(0) = \lambda$. Use that $G(0) = 1$.

h.1.17.7 Substitute the recursion, and carry on with the algebra. We urge the reader to try it, its good (necessary) practice. Use also the results of the exercises of Section 1.15.

h.1.17.8 Divide both sides by μ.

h.1.17.9 This is an important problem, it helps to check (numerically) the algebraic results we have been deriving up to now. Implementing the recursion is not hard, just try and see how far you get.

h.1.17.10 This exercise tests your creativity and modeling skills, it is not analytically difficult. The best approach to problems like this is to try some simple cases first. For instance, consider the case with batch sizes of 1 first, then batches of sizes 1 or 2, and so on. If system contains K jobs, which batches can be accepted? If the system contains $K-3$, say, which batch sizes can be accepted, which will be refused, under which acceptance policy?

Solutions

s.1.17.1 $A(n,n,t)$ counts all jobs up to time t that see n items and bring at least $n-n+1=1$ unit of work. As each job brings at least 1 item, $A(n,n,t)$ counts all jobs that see n items at arrival.

s.1.17.2 Realize that B and $L(A-)$ are assumed to be independent.

$$\begin{aligned}\mathsf{P}(B>n-m\,|\,L(A-)=m) &= \frac{\mathsf{P}(B>n-m, L(A-)=m)}{\mathsf{P}(L(A-)=m)} \\ &= \frac{\mathsf{P}(B>n-m)\mathsf{P}(L(A-)=m)}{\mathsf{P}(L(A-)=m)} = \mathsf{P}(B>n-m).\end{aligned}$$

s.1.17.3 Eq. (1.83) has the interpretation that the rate at which level n is crossed from below from state m is equal the rate at which jobs arrive times the fraction of jobs that see m jobs in the system times the fraction of jobs with batch size larger than $n-m$. Observe that the stream of jobs with batch size larger than $n-m$ is a Poisson process thinned at rate $G(n-m)$.

s.1.17.4 The left hand side of Eq. (1.85) is identical, so we only have to concentrate on the right hand side. In the $M/M/1$ queue, all batches have size 1. Thus, $\mathsf{P}(B=1)=f(1)=1$ and $f(k)=0$ for $k\neq 1$. Thus, $G(0)=1$ and $G(1)=G(2)=\cdots=0$. Thus, $\sum_{m=0}^{n} G(n-m)p(m)=G(0)p(n)=p(n)$.

s.1.17.5

$$\begin{aligned}\pi(1) &= \alpha \\ \pi(2) &= \alpha G(0)\pi(1)+\alpha G(1)\pi(0)=\alpha(\alpha+G(1))=\alpha^2+\alpha G(1), \\ \pi(3) &= \alpha[G(0)\pi(2)+G(1)\pi(1)+G(2)\pi(0)]=\alpha[\alpha^2+2\alpha G(1))+G(2)],\end{aligned}$$

s.1.17.6 Because arrivals do not occur in single units, but in batches.

s.1.17.7 We use that $\mu\pi(n) = \lambda\sum_{i=0}^{n-1}\pi(i)G(n-1-i)$ and the results of the exercises of Section 1.15 to see that

$$\begin{aligned}
\mu\,\mathsf{E}[L] &= \sum_{n=0}^{\infty} n\,\mu\pi(n), \quad \text{now substitute for } \mu\pi(n) \text{ the above recursion,}\\
&= \lambda\sum_{n=0}^{\infty} n\sum_{i=0}^{n-1}\pi(i)G(n-1-i) = \lambda\sum_{n=0}^{\infty} n\sum_{i=0}^{\infty}1\{i<n\}\pi(i)G(n-1-i)\\
&= \lambda\sum_{i=0}^{\infty}\pi(i)\sum_{n=0}^{\infty}1\{i<n\}nG(n-1-i) = \lambda\sum_{i=0}^{\infty}\pi(i)\sum_{n=i+1}^{\infty}nG(n-1-i)\\
&= \lambda\sum_{i=0}^{\infty}\pi(i)\sum_{n=0}^{\infty}(n+i+1)G(n) = \lambda\sum_{i=0}^{\infty}\pi(i)\left[\sum_{n=0}^{\infty}nG(n)+(i+1)\sum_{n=0}^{\infty}G(n)\right]\\
&= \lambda\sum_{i=0}^{\infty}\pi(i)\sum_{n=0}^{\infty}nG(n)+\lambda\,\mathsf{E}[B]\sum_{i=0}^{\infty}\pi(i)(i+1)\\
&= \lambda\sum_{i=0}^{\infty}\pi(i)\frac{\mathsf{E}[B]^2-\mathsf{E}[B]}{2}+\lambda\,\mathsf{E}[B](\mathsf{E}[L]+1)\\
&= \lambda\frac{\mathsf{E}[B]^2-\mathsf{E}[B]}{2}+\lambda\,\mathsf{E}[B]\,\mathsf{E}[L]+\lambda\,\mathsf{E}[B]\\
&= \lambda\frac{\mathsf{E}[B]^2}{2}+\lambda\,\mathsf{E}[B]\,\mathsf{E}[L]+\lambda\frac{\mathsf{E}[B]}{2},
\end{aligned}$$

s.1.17.8 Dividing both sides by μ and using that $\lambda\,\mathsf{E}[B]/\mu = \rho$,

$$\mathsf{E}[L] = \frac{\lambda}{\mu}\frac{\mathsf{E}[B]^2}{2}+\rho\,\mathsf{E}[L]+\frac{\rho}{2}.$$

s.1.17.9 The recursion for n becomes

$$\pi(n) = \frac{\lambda}{\mu}\sum_{i=0}^{n-1}\pi(n-1-i)G(i).$$

Since $G(i) = 0$ for $i \geq 3$ we rewrite this to

$$\pi(n) = \frac{\lambda}{\mu}\sum_{i=0}^{\min(\{n-1,l\})}\pi(n-1-i)G(i),$$

where $l = 3$.

In python this reads:

```
p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, l)))
```

The following code carries out the recursion for the $M/M/1$ queue and compares the result to the *unnormalized* probabilities $(\lambda/\mu)^n$ (unnormalized means that I do not multiply with $1-\rho$). Thus, we can test the code right away.

```
>>> import numpy as np

>>> l = 1 # M/M/1 has batch size 1
>>> f = np.ones(l + 1) / l # f[0] =1, f[1] = 1
>>> f[0] = 0 # set f[0] = 0
>>> F = np.cumsum(f) # distribution
>>> G = np.ones_like(F) - F # survivor function

>>> labda = 1
>>> mu = 3
>>> num = 30
>>> p = np.ones(num)

>>> for n in range(1, num):
...     p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n
...     print(n, p[n], (labda / mu)**n)
...
1 0.3333333333333333 0.3333333333333333
2 0.1111111111111111 0.1111111111111111
3 0.037037037037037035 0.03703703703703703
4 0.012345679012345678 0.012345679012345677
5 0.004115226337448559 0.004115226337448558
6 0.0013717421124828853 0.0013717421124828527
7 0.0004572473708276176 0.00045724737082761756
8 0.00015241579027587253 0.000152415790275872 5
9 5.080526342529084e-05 5.0805263425290837e-05
10 1.693508780843028e-05 1.693508780843028e-05
11 5.64502926947676e-06 5.645029269476759e-06
12 1.8816764231589197e-06 1.8816764231589195e-06
13 6.272254743863065e-07 6.272254743863065e-07
14 2.0907515812876884e-07 2.090751581287688e-07
15 6.969171937625627e-08 6.969171937625627e-08
16 2.3230573125418757e-08 2.3230573125418753e-08
```

```
17 7.743524375139585e-09 7.743524375139585e-09
18 2.581174791713195e-09 2.581174791713195e-09
19 8.603915972377316e-10 8.603915972377315e-10
20 2.867971990792439e-10 2.867971990792438e-10
21 9.559906635974796e-11 9.559906635974793e-11
22 3.186635545324932e-11 3.186635545324931e-11
23 1.0622118484416439e-11 1.0622118484416435e-11
24 3.540706161472146e-12 3.540706161472145e-12
25 1.180235387157382e-12 1.1802353871573816e-12
26 3.9341179571912734e-13 3.934117957191272e-13
27 1.311372652397091e-13 1.3113726523970905e-13
28 4.3712421746569697e-14 4.3712421746569684e-14
29 1.4570807248856565e-14 1.457080724885656e-14
```

The test is convincing.

```
>>> l = 3
>>> f = np.ones(l + 1) / l
>>> f[0] = 0
>>> F = np.cumsum(f)
>>> G = np.ones_like(F) - F
>>> num = 30
>>> p = np.ones(num)
>>> for n in range(1, num):
...     p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n
...     print(n, p[n])
...
1 0.3333333333333333
2 0.3333333333333333
3 0.2962962962962963
4 0.20987654320987653
5 0.17283950617283952
6 0.13717421124828533
7 0.10745313214449018
8 0.08550525834476452
9 0.06762180561906214
10 0.053481007299022845
11 0.04235465460888414
12 0.033516420449306694
```

```
13 0.02652661976274569
14 0.020996372755081625
15 0.016617642026671438
16 0.013152476816991472
17 0.010409898584377658
18 0.008239143490420816
19 0.006521078273001026
20 0.0051612689315802636
21 0.004085011870129295
22 0.0032331835273943822
23 0.002558982583780222
24 0.0020253696306954143
25 0.001603028620782341
26 0.0012687564119464527
27 0.001004188456455495
28 0.0007947896460046368
29 0.0006290558069857058
```

Stopping at 30 seems ok. The last step is normalize.

```
>>> p /= p.sum()  # normalize
>>> for n in range(1, num):  # print normalized results
...     print(n, p[n])
...
1 0.11119961231552707
2 0.11119961231552707
3 0.09884409983602406
4 0.07001457071718371
5 0.05765905823768071
6 0.04576115733149262
7 0.035846239909669224
8 0.028524454736630404
9 0.022558555706746922
10 0.017841201833685642
11 0.014129463516797907
12 0.011181038880501324
13 0.008849249500976158
14 0.007004365531192115
15 0.00554362605289222
```

```
16 0.004387650969115228
17 0.003472730060480249
18 0.00274856868584038
19 0.002175424127510763
20 0.0017217933127436998
21 0.0013627552087881116
22 0.0010785862643736108
23 0.0008536736137156394
24 0.0006756609531869169
25 0.0005347684834850711
26 0.0004232556633938541
27 0.00033499610114873577
28 0.0002651409015243318 3
29 0.0002098522855849245 3
```

Now that we have the probabilities we can do all experiments we like.

```
>>> EL = sum(n*p[n] for n in range(len(p)))
>>> EL
3.309069824793065
```

It is of interest to compare this result to the equation above Eq. (1.76).

```
>>> EB = sum(k*fk for k,fk in enumerate(f))
>>> EB2 = sum(k*k*fk for k,fk in enumerate(f))
>>> EB2
4.666666666666666
>>> rho=labda*EB/mu
>>> RHS = labda/mu*EB2/2 + rho/2
>>> EL=RHS/(1-rho)
>>> EL
3.3333333333333326
```

So, after all, stopping at 30 seems not quite ok: there is a slight difference between the two expectations. Lets run the recursion up to 100, and see what we get then.

```
>>> num = 100
>>> p = np.ones(num)

>>> for n in range(1, num):
```

```
...     p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n
...
>>> p /= p.sum()  # normalize
>>> EL = sum(n*p[n] for n in range(len(p)))
>>> print(EL)
3.3333333271183934
```

This does the job.

As a last step, I want to check Eq. (1.76). In fact, I spent at least one hour to get Eq. (1.76) correct. Now I can also numerically check it.

```
>>> VB = EB2 - EB*EB
>>> C2 = VB/EB/EB
>>> EL = (1+C2)/2*rho/(1-rho)*EB + rho/(1-rho)/2
>>> EL
3.3333333333333326
```

The results agree.

s.1.17.10 Let's first deal with complete rejection. Suppose a batch of size k arrives when the system contains n jobs. When $k+n \le K$, the batch can be accepted since the entire batch will fit into the queue. When, however, $k+n > K$, the batch has to be rejected.

Now consider an imaginary line between states with n and $n+1$ jobs in the system. This imaginary line separates the state space into two disjoint part: one part with states $0,1,\ldots,n$ and the other part with states $n+1,n+2,\ldots,K$. We call this imaginary line 'level n'. We call an 'upcrossing' a transition from some state $m \le n$ to a state $l \ge n+1$. We also say in that case that level n is 'crossed from below', i.e., from some state $m \le n$ to a state $l \ge n+1$. Likewise, a 'downcrossing' of level n happens when this level is crossed from 'above', i.e. from some state $l \ge n+1$ to some state $m \le n$.

If the system contains n jobs, level n is crossed from below with rate $\lambda\pi(n)\mathsf{P}(B \le K-n$ More generally, when the system is in state $m \le n$, we need a batch of at least $n+1-m$ to cross the n. Moreover, any batch larger than $K+1-m$ gets rejected. Thus, when the system contains $m \le n$ jobs, the rate at which level n is crossed from below is

$$\lambda\pi(m)\mathsf{P}(n+1-m \le B \le K+1-m) = \lambda\pi(m)[G(n-m)-G(K-m)].$$

Since the server serves only single items, level n can only be crossed from above from state $n+1$. This happens at rate $\mu p(n+1)$.

Finally, since on the long run the number of up- and crossings must be the same, the up- and downcrossing rates must match. This implies that the balance equations becomes

$$\mu\pi(n+1) = \lambda \sum_{m=0}^{n} \pi(m)[G(n-m) - G(K-m)],$$

for $n = 0, \ldots, K-1$.

Before we continue with the other acceptance rules, it is important to check this result. In general, off-by-one errors are easily, and commonly, made, so we need to test the above on simple cases.

- If $K \to \infty$, then $G(K-m) = \mathsf{P}(B > K-m) \to 0$, so we get our earlier result.
- Take $n = K$. Then $G(n-m) - G(K-m) = 0$ for all m. Then the right hand side is 0, as it should.
- Taken $n = 0$ and $K = 1$, then $\mu\pi(1) = \lambda\pi(0)$. This also makes sense.
- Take n much smaller than K. If the batch size is maximally 2, then for small n the entire batch must fit. Let's see if this holds in the above formula. If n much smaller than K, then also m is much smaller than K (since in the right hand size, $m \le n$. But then $G(K-m) \le G(2) = 0$, as it should. (Observe that G is a decreasing function of it argument; its a survival function.)

The complete-acceptance policy is actually quite simple. As any batch will be accepted when $n \le K$, the queue length is not bounded. Only when the number of jobs in the system is larger than K, we do not accept jobs.

$$\mu\pi(n+1) = \begin{cases} \lambda \sum_{m=0}^{n} \pi(m) G(n-m), & \text{for } n \le K, \\ \lambda \sum_{m=0}^{K} \pi(m) G(n-m), & \text{for } n > K. \end{cases}$$

For the partial acceptance case, any job is accepted, but the system only admits whatever fits. Thus, the rate at which level $n < K$ is crossed is not affected by this acceptance policy, and therefore,

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

for $n = 0, 1, \ldots, K-1$.

1.18 M/G/1 Queue Length Distribution

Theory and Exercises

In Section 1.17 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', which as we will see 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 relation (1.64) 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 for an i.i.d. sequence of random variables, the sequence $\{Y_k\}$ is 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.

1.18.1. Explain that if the service time is constant and equal to s, then

$$\mathsf{P}(Y_k = j \,|\, S = s) = e^{-\lambda s} \frac{(\lambda s)^j}{j!}. \tag{1.88}$$

1.18.2. Explain that

$$\mathsf{P}(Y_k = j) = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!} \, \mathrm{d}F(x). \tag{1.89}$$

where F is the distribution of the service times.

1.18.3. If S is deterministic and equal to s, show that Eq. (1.89) reduces to Eq. (1.88).

1.18.4. If $S \sim \exp(\mu)$, show that

$$f(j) = \mathsf{P}(Y_k = j) = \frac{\mu}{\lambda+\mu}\left(\frac{\lambda}{\lambda+\mu}\right)^j. \tag{1.90}$$

1.18.5. If $S \sim \exp(\mu)$, show that

$$G(j) = \sum_{k=j+1}^{\infty} f(k) = \left(\frac{\lambda}{\lambda+\mu}\right)^{j+1}. \tag{1.91}$$

Let us concentrate on a downcrossing of level n, see Figure 1.22; recall that level n lies between states n and $n+1$. For job k to generate a downcrossing 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,

$$\text{Downcrossing of level } n \iff \mathbb{1}[L(D_{k-1}) = n+1]\,\mathbb{1}[L(D_k) = n] = 1.$$

Let us write this in another way. Observe that if $L(D_{k-1}) = n+1$ and no other jobs arrive during the service time S_k of job k, i.e., when $Y_k = 0$, it must also be that job k leaves n jobs behind. If, however, $Y_k > 0$, then $L(D_k) \geq n+1$. Thus, we see that

$$\text{Downcrossing of level } n \iff \mathbb{1}[L(D_{k-1}) = n+1]\,\mathbb{1}[Y_k = 0] = 1.$$

Consequently, the number of downcrossings of level n up to time t is

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

1.18.6. Use a similar derivation as in (1.84) to 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 upcrossing, it is important to do the next exercise.

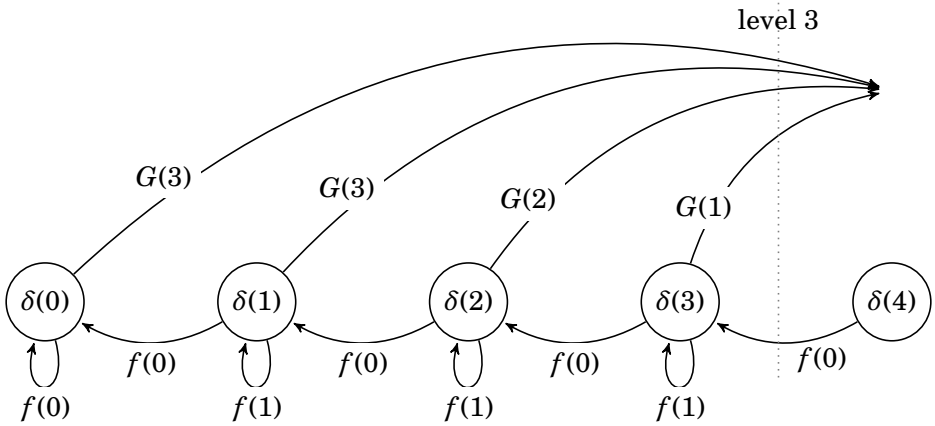


Figure 1.22: Level 3 is crossed from below with rate $\delta\delta(0)G(3)+\delta\delta(1)G(3)+\cdots\delta\delta(3)G$ and crossed from above with rate $\delta\delta(4)f(0)$.

1.18.7. Suppose that $L(D_{k-1})>0$. Why is $S_k = D_k - D_{k-1}$? 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 distribution of $D_k - D_{k-1}$ in case $L(D_{k-1}) = 0$?

For the upcrossings, assume first that $L(D_{k-1}) = n > 0$. Then an upcrossing of level $n > 0$ must have occurred when $L(D_k) > n$, i.e,

$$\mathbb{1}[L(D_{k-1}) = n]\,\mathbb{1}[L(D_k) > n] = 1 \implies \text{Upcrossing of level } n.$$

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 upcrossing can happen. Next, if $Y_k = 1$, then job k leaves n jobs behind, so still no upcrossing occurs. In fact, level n can only be upcrossed from level n if more than one job arrives during the service of job k, i.e.,

$$\mathbb{1}[L(D_{k-1}) = n]\,\mathbb{1}[Y_k > 1] \implies \text{Upcrossing of level } n.$$

More generally, level n is upcrossed from level m, $0 < m \le n$ whenever

$$\mathbb{1}[L(D_{k-1}) = m]\,\mathbb{1}[Y_k > n-m+1] \implies \text{Upcrossing of level } n.$$

However, if $m = 0$ (think about this),

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

Again we define proper counting functions, divide by t, and take suitable limits to find for upcrossing rate

$$\delta\delta(0)G(n)+\delta\sum_{m=1}^{n}\delta(m)G(n-m+1). \tag{1.92}$$

1.18.8. Provide the details behind the derivation of Eq. 1.92.

Equating the downcrossing and upcrossing 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 $\pi(n)=\delta(n)$, which follows from (1.64) and the fact that the $M/G/1$ queue length process has one-step transitions, we arrive at

$$f(0)\pi(n+1)=\pi(0)G(n)+\sum_{m=1}^{n}\pi(m)G(n+1-m). \tag{1.93}$$

Clearly, we have again obtained a recursion by which we can compute, iteratively, the state probabilities, and follow the approach sketched below Exercise 1.17.5.

1.18.9. Clearly, the $M/M/1$ queue is a special case of the $M/G/1$ queue. Apply Eq. (1.93) to the $M/M/1$ queue to obtain $p(n)=(1-\rho)\rho^n$.

1.18.10. Can you design a suitable numerical method to evaluate Eq. (1.89) for more general distribution functions F? (This topic is not strictly necessary for this course, but I hope that you study it nonetheless.)

Hints

h.1.18.1 If s is deterministic, the number of arrival during a fixed period of time with length s must be Poisson distributed.

h.1.18.3 $S=s$ then $\mathsf{P}(S=s)=1$, i.e., all probability mass lies at s. Thus, all arrivals must occur during $[0,s]$.

h.1.18.7 Realize that if $L(D_{k-1}) = 0$, job $k-1$ leaves behind an empty system. Thus, before job k can leave, it has to arrive. In other words, $D_{k-1} < A_k$. Since job k arrives to an empty system, his service starts right away, to that the time between A_k and D_k is equal to the service time of job k.

h.1.18.8 Define for $m = 1, \ldots, n$

$$D(m,n,t) = \sum_{k=1}^{D(t)} \mathbb{1}[L(D_{k-1}) = m]\,\mathbb{1}[Y_k > n - m + 1],$$

and

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

Then, divide by $D(n,t)$ and $D(t)$ and take limits.

h.1.18.9 Define shorthands such as $\alpha = \lambda/(\lambda+\mu)$, so that $1-\alpha = \mu/(\lambda+\mu)$, and $\alpha/(1-\alpha) = \lambda/\mu = \rho$. Then, with Eq. 1.90, $f(n) = \alpha^n(1-\alpha)$ and $G(n) = \alpha^{n+1}$. (These results do not 'come for 'free'. Of course, it's just algebra, but please try to derive this yourself. Its good to hone your computational skills.)

h.1.18.10 Discretize time to a grid of points, and approximate the integral by a summation over the grid.

Solutions

s.1.18.1 See the hint. The period during which the arrivals occur is s.

s.1.18.2 We use a conditioning argument to arrive at this result. The probability that the service time is x units long is written in various ways in the literature: $F(\mathrm{d}x) = \mathrm{d}F(x) = \mathsf{P}(S \in \mathrm{d}x)$, but this all means the same thing, it is only the notation that differs. (As an aside, to properly define this we need measure theory). When F has a density f, then $\mathrm{d}F(x) = f(x)\mathrm{d}x$. Note that when S is discrete, it does not have a density everywhere. With this,

$$\mathsf{P}(Y_k = j) = \int_0^\infty \mathsf{P}(Y_k = j \,|\, S = x)\,\mathsf{P}(S \in \mathrm{d}x) = \int_0^\infty \mathsf{P}(Y_k = j \,|\, S = x)\,\mathrm{d}F(x).$$

Using the answer of the previous problem we arrive at the result.

s.1.18.3 Let us first attack this problem from a general point of view. Suppose the service time S can take values $s_1 < s_2 < \cdots < s_n$, and $\mathsf{P}(S = s_i) = \alpha_i$. Then of course we want that $\mathsf{P}(S \le s_n) = \sum_{i=1}^{n} \alpha_i = 1$. The distribution function F of S is in this case a step function, with steps at the points $s_1, s_2, \ldots$, and step sizes $\alpha_1, \alpha_2, \ldots$. Thus, $F(s_i) - F(s_i-) = \alpha_i$. We say that such a distribution function has *atoms* at the points $s_1, s_2, \ldots$. In this case we write

$$\int_0^\infty g(x)\,\mathrm{d}F(x) = \sum_{i=1}^{n} g(s_i)\alpha_i.$$

Thus, the integral of g with respect to F is the sum of g at the points at which F makes a jump times the weight of F at these points.

As an example, in case $S \equiv 10$ (the service time is always 10 time units long) the distribution function F makes just one jump at $s_1 = 10$ of size $\alpha_1 = F(10) - F(10-) = 1$, i.e., F has the form

$$F(x) = \begin{cases} 0, & x < 10, \\ 1, & x \ge 10. \end{cases}$$

With this,

$$\int_0^\infty g(x)\,\mathrm{d}F(x) = \sum_{i=1}^{n} g(s_i)\alpha_i = g(s_1)\alpha_1 = g(10)\cdot 1 = g(10).$$

Thus, the integral of g with respect to this distribution F is $g(10)$. More generally, when F puts all probability mass at the single point s (rather than at the point 10), then

$$\int_0^\infty g(x)\,\mathrm{d}F(x) = g(s).$$

Let us now copy the formula of the previous problem:

$$\mathsf{P}(Y_k = j) = \int_0^\infty \mathsf{P}(Y_k = j \,|\, S = x)\,\mathrm{d}F(x).$$

We see that here the integrand is the function $g(x) = \mathsf{P}(Y_k = j \,|\, S = x)$. It is given in the question that F puts all mass on the point s. Therefore,

$$\mathsf{P}(Y_k = j) = \int_0^\infty \mathsf{P}(Y_k = j \,|\, S = x)\,\mathrm{d}F(x) = \int_0^\infty g(x)\,\mathrm{d}F(x) = g(s) \qquad \star$$

Now we also know that if the service time takes precisely x time units, the number of arrivals is Poisson distributed. Therefore

$$\mathsf{P}(Y_k = j \,|\, S = x) = e^{-\lambda x}\frac{(\lambda x)^j}{j!}.$$

Hence,

$$g(x) = \mathsf{P}(Y_k = j \,|\, S = x) = e^{-\lambda x}\frac{(\lambda x)^j}{j!}.$$

Finally, using ($\star$), and taking $x = s$ in the above expression of g, we get

$$\mathsf{P}(Y_k = j) = \int_0^\infty \mathsf{P}(Y_k = j \,|\, S = x)\, \mathrm{d}F(x) = g(s) = e^{-\lambda s}\frac{(\lambda s)^j}{j!}.$$

s.1.18.4 Use conditional probability to see that

$$\begin{aligned}
\mathsf{P}(Y_n = j) &= \int_0^\infty e^{-\lambda x}\frac{(\lambda x)^j}{j!}\,\mathrm{d}F(x) = \int_0^\infty e^{-\lambda x}\frac{(\lambda x)^j}{j!}\mu e^{-\mu x}\,\mathrm{d}x \\
&= \frac{\mu}{j!}\lambda^j\int_0^\infty e^{-(\lambda+\mu)x}x^j\,\mathrm{d}x = \frac{\mu}{j!}\left(\frac{\lambda}{\lambda+\mu}\right)^j\int_0^\infty e^{-(\lambda+\mu)x}((\lambda+\mu)x)^j\,\mathrm{d}x \\
&= \frac{\mu}{j!}\left(\frac{\lambda}{\lambda+\mu}\right)^j\frac{j!}{\lambda+\mu}.
\end{aligned}$$

Here I also used the hints and results of Exercise 1.2.7.

In hindsight, this result could have been derived in another way, in fact by using the result of Exercise 1.2.17 in which we analyzed a merged Poisson process. Consider the Poisson process with rate $\lambda+\mu$ that arises when the arrival and service process are merged. The probability that an arrival corresponds to an epoch of the merged process is $\lambda/(\lambda+\mu)$ and the probability that a departure corresponds to an epoch of the merged process is $\mu/(\lambda+\mu)$. The probability that j arrivals occur before a service occurs, is the same as the probability that a geometrically distributed random variable with success probability $\mu/(\lambda+\mu) = 1-p$ takes the value j.

s.1.18.5 Take $\alpha = \lambda/(\lambda+\mu)$ so that $f(j) = (1-\alpha)\alpha^j$.

$$\begin{aligned}
G(j) &= \sum_{k=j+1}^\infty f(k) = (1-\alpha)\sum_{k=j+1}^\infty \alpha^k \\
&= (1-\alpha)\sum_{k=0}^\infty \alpha^{k+j+1}, \text{ by change of variable} \\
&= (1-\alpha)\sum_{k=0}^\infty \alpha^k\alpha^{j+1} = (1-\alpha)\alpha^{j+1}\sum_{k=0}^\infty \alpha^k \\
&= (1-\alpha)\alpha^{j+1}\frac{1}{1-\alpha} = \alpha^{j+1}.
\end{aligned}$$

s.1.18.6 By using the definitions and limits developed in Sections 1.7 and 1.12,

$$\begin{aligned}\lim_{t\to\infty}\frac{D(n+1,0,t)}{t} &= \lim_{t\to\infty}\frac{D(t)}{t}\frac{D(n+1,t)}{D(t)}\frac{D(n+1,0,t)}{D(n+1,t)}\\ &= \delta\delta(n+1)\lim_{t\to\infty}\frac{D(n+1,0,t)}{D(n+1,t)}\\ &= \delta\delta(n+1)\mathsf{P}(Y=0)\\ &= \delta\delta(n+1)f(0),\end{aligned}$$

where the last limit follows from the independence of Y_k and $L(D_{k-1})$,

s.1.18.7 When $L(D_{k-1})>0$, job k is already in the system when job $k-1$ finishes its service and leaves. Thus, at the departure time D_{k-1} of job $k-1$, the service of job k can start right away at D_{k-1}. Then, $D_k = D_{k-1}+S_k$.

When job $k-1$ leaves an empty system behind, $D_k = A_k + S_k$, since job k sees an empty system, hence its service can start right away after its arrival time at time A_k. Since the arrival process is Poisson by assumption, the time to the next arrival after D_{k-1} is exponentially distributed with rate λ. (Recall the memoryless property of the interarrival times.) Thus, $A_k - D_{k-1}$ has the same distribution as X_k, so that $\mathsf{P}(D_k - D_{k-1}\le x) = \mathsf{P}(X_k + S_k \le x)$. BTW: Can you simplify this expression by using conditioning on X_k?

s.1.18.8 With the definition of the hint,

$$\begin{aligned}\lim_{t\to\infty}\frac{D(0,n,t)}{t} &= \lim_{t\to\infty}\frac{D(t)}{t}\frac{D(0,t)}{D(t)}\frac{D(0,n,t)}{D(0,t)}\\ &= \delta\delta(0)\lim_{t\to\infty}\frac{D(0,n,t)}{D(0,t)}\\ &= \delta\delta(0)\mathsf{P}(Y>n)\\ &= \delta\delta(0)G(n).\end{aligned}$$

$$\begin{aligned}\lim_{t\to\infty}\frac{D(m,n,t)}{t} &= \lim_{t\to\infty}\frac{D(t)}{t}\frac{D(m,t)}{D(t)}\frac{D(m,n,t)}{D(m,t)}\\ &= \delta\delta(m)\lim_{t\to\infty}\frac{D(m,n,t)}{D(m,t)}\\ &= \delta\delta(m)\mathsf{P}(Y>n-m+1)\\ &= \delta\delta(m)G(n-m+1).\end{aligned}$$

s.1.18.9 Take $n = 0$. Then $f(0)\pi(1) = (1-\alpha)\pi(1)$, and $\pi(0)G(0) = \pi(0)\alpha$. Thus, $\pi(1) = \pi(0)\alpha/(1-\alpha) = \rho\pi(0)$.

For $n = 1$,

$$\begin{aligned}(1-\alpha)\pi(2) &= \pi(0)G(1) + \pi(1)G(1) = \pi(0)G(1)(1+\rho)\\ &= \pi(0)\alpha^2(1+\rho) = \pi(0)\alpha\alpha(1+\rho) = \pi(0)\alpha\rho.\end{aligned}$$

Dividing by $1-\alpha$, we get

$$\pi(2) = \pi(0)\rho^2$$

Finally, now that we suspect that $\pi(n) = \rho^n\pi(0)$, let's fill it in, and see whether it checks. We divide both sides by $\pi(0)$ so that we are left with checking that

$$\begin{aligned}(1-\alpha)\rho^{n+1} &= \alpha^{n+1} + \sum_{m=1}^{n}\rho^m\alpha^{n-m+2}\\ &= \alpha^{n+1} + \alpha^{n+2}\sum_{m=1}^{n}(\rho/\alpha)^m\\ &= \alpha^{n+1} + \alpha^{n+1}\rho\sum_{m=0}^{n-1}(\rho/\alpha)^m\\ &= \alpha^{n+1} + \alpha^{n+1}\rho\sum_{m=0}^{n-1}(\rho/\alpha)^m\\ &= \alpha^{n+1} + \alpha^{n+1}\rho\frac{1-(\rho/\alpha)^n}{1-\rho/\alpha)}\\ &= \alpha^{n+1} - \alpha^{n+1}(1-(\rho/\alpha)^n), \quad \text{as } \rho/\alpha = 1+\rho,\\ &= \alpha^{n+1}(\rho/\alpha)^n = \alpha\rho^n.\end{aligned}$$

Since $\rho = \alpha/(1-\alpha)$ we see that the left and right hand sides are the same.

Thus we get that $\pi(n) = \delta(n) = (1-\rho)\pi(0)$; a result we obtained earlier for the $M/M/1$ queue.

s.1.18.10 A simple numerical method is as follows. Make a grid of size $\mathrm{d}x$, for some small number $\mathrm{d}x$, e.g. $\mathrm{d}x = 1/100$, and write $f_i = \mathsf{P}(S \in (i\,\mathrm{d}x, (i+1)\,\mathrm{d}x]) = F((i+1)\,\mathrm{d}x) - F(i\,\mathrm{d}x)$. Then

$$\mathsf{P}(Y_k = j) = \int_0^\infty e^{-\lambda s}\frac{(\lambda x)^j}{j!}\,\mathrm{d}F(x) \approx \sum_{i=1}^{\infty} e^{-\lambda i\,\mathrm{d}x}\frac{(\lambda i\,\mathrm{d}x)^j}{j!}f_i\,\mathrm{d}x$$

Lets try a numerical experiment.

```
>>> import numpy as np

>>> labda = 3
>>> mu = 4
>>> j = 5
>>> dx = 1 / 100


>>> def F(x):
...     return 1 - np.exp(-mu * x)
...

>>> def f(x):
...     return F(x + dx) - F(x)
...

>>> def term(i):
...     res = np.exp(-labda * i * dx)
...     res *= (labda * i * dx)**j / np.math.factorial(j)
...     res *= f(i*dx) * dx
...     return res
...
>>> print(sum(term(i) for i in range(50)))
1.1158835007226524e-05
>>> print(sum(term(i) for i in range(500)))
8.098807710055904e-05
>>> print(sum(term(i) for i in range(5000)))
8.098807712728956e-05
```

Since I don't know when to stop the integral I just try a few values; of course stopping the integration at $x = 50$ is too small, since $50\,\mathrm{d}x = 50/100 = 1/2$, but I include it for illustrative purposes. Stopping at 500 seems ok, since the results of the last two integrals are nearly the same. This also suggeststst that $\mathrm{d}x = 1/100$ is sufficiently small. In general, however, one must take care and try various values for $\mathrm{d}x$ and the integration limits.

For more complicated situations it is best to use a numerical library to compute the above integral. These methods have been designed to produce good and reliable results, and, typically, it is very hard to improve these methods. Thus, let's try a real number cruncher.

```
>>> from scipy.integrate import quad

>>> def g(x):
...     return np.exp(-labda*x) * (labda*x)**j/np.math.factorial(j) * f(
...
>>> print(quad(g, 0, np.inf))
(8.098807712760667e-05, 3.4086429822163874e-11)
```

This is the same as our earlier answer.

1.19 G/G/1 and G/G/c Queue: Approximations

Theory and Exercises

In manufacturing settings it is quite often the case that the arrival process at a station is not Poisson. 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 interarrival times at the second station must be more or less equal. 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 consequence the Pollackzek-Khintchine formula for the $M/G/1$-queue can no longer be reliably used to compute the average waiting times. As a second, trivial case, if the interarrival times of jobs are 1 hour always and service times 59 minutes always, there simply cannot be queue. Thus, the $M/G/1$ waiting time formula cannot, and should not, be applied to approximate the average waiting time of the $G/G/1$ queue.

There is no formula as yet by which the average waiting times for the $G/G/1$ queue can be computed; only approximations are available. One such simple and robust approximation is based on the following observation. Recall the waiting time in queue for the $M/M/1$ queue:

$$\mathsf{E}\left[W_Q(M/M/1)\right] = \frac{\rho}{1-\rho}\mathsf{E}[S] = \frac{1_a + 1_b}{2}\frac{\rho}{1-\rho}\mathsf{E}[S],$$

where we label the number 1 with a and b. When generalizing this result to the $M/G/1$ queue we get

$$\mathsf{E}\left[W_Q(M/G/1)\right] = \frac{1_a + C_s^2}{2}\frac{\rho}{1-\rho}\mathsf{E}[S],$$

Thus, 1_b in the expression for the $M/M/1$ queue is replaced by C_s^2 in the expression for the $M/G/1$ queue. As a second generalization, Kingman proposed to replace 1_a in

this formula by the SCV of the interarrival times

$$C_a^2 = \frac{\mathsf{V}[X]}{(\mathsf{E}[X])^2}$$

resulting in

$$\mathsf{E}\left[W_Q(G/G/1)\right] \approx \frac{C_a^2 + C_s^2}{2} \frac{\rho}{1-\rho} \mathsf{E}[S].$$

This formula is reasonably accurate; for related expressions we refer to Bolch et al. [2006] and Hall [1991]. With Little's law we can compute $\mathsf{E}\left[L_Q\right]$ from the above. Moreover, $\mathsf{E}[W] = \mathsf{E}\left[W_Q\right] + \mathsf{E}[S]$, and so on, c.f., Section 1.14

It is crucial to memorize the *scaling* relations that can be seen from the $G/G/1$ waiting time formula. 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 is very sensitive to the actual value of ρ when ρ is large.

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

3. $\mathsf{E}\left[W_Q\right] \sim \mathsf{E}[S]$. Thus, working in smaller job sizes reduces the waiting time also. The average queue length does not decrease by working with smaller batches, but jobs are more 'uniformly spread' over the queue. This effect lies behind the idea of 'lot-splitting', i.e., rather than process large jobs, split jobs into multiple small jobs (assuming that setup times are negligible), so that the waiting time per job can be reduced.

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.

For the $G/G/c$ queue we can Sakasegawa's approximation, Sakasegawa [1977],

$$\mathsf{E}\left[W_Q\right] = \frac{C_a^2 + C_e^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} \mathsf{E}[S] \tag{1.94}$$

to estimate the time in queue. We refer to Hopp and Spearman [2008] for a discussion of this formula and its many applications.

Even though the above results are only approximate, 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.

1.19.1. ($G/G/1$) Show that the approximation (1.94) reduces to the result known for the $M/M/1$ and $M/G/1$ queues.

1.19.2. Is Eq. (1.67) also valid for the $G/G/1$ queue? Why (not)?

1.19.3. Consider a queue n servers, with generally distributed interarrival times, generally distributed service times, and the system can contain at most K customers, i.e, the $G/G/n/K$ 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

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

1.19.4. Consider a single-server queue at which every minute a customer arrives, precisely at the first second. Each customer requires precisely 50 seconds of service. What are ρ, $\mathsf{E}[L]$, C_a^2, and C_S^2?

1.19.5. Consider the same single-server system as in the previous exercise, 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.

1.19.6. For the $G/G/1$ queue, prove that the fraction of jobs that see n jobs in the system is the same as the the fraction of departures that leave n jobs behind. What condition have you used to prove this?

1.19.7. Suppose for the $G/G/1$ that a job sees n jobs in the system upon arrival. Can you use the central limit law to estimate the distribution of waiting time in queue for this job?

1.19.8. (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 interarrival 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, Kingman's equation requires an estimate of the SCV C_a^2 of the interarrival 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 information that is available. However, when for instance the number of arrivals per day have 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 the interarrival 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 1.19.1. The SCV of the interarrival times can be estimated with the following 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 1.7. 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 interarrival 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 uses a similar relation in the derivation of the Poisson process):

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

Since the interarrival 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 $\mathsf{P}(N(0,1) \le y) = \mathsf{P}(N(0,1) > -y)$ (and $\mathsf{P}(N(0,1) = y) = 0$) we have that

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

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,

$$\Phi(y) \approx \mathsf{P}(A_k > t_k)$$

$$= \mathsf{P}(A(t_k) < k)$$
$$= \mathsf{P}\left(\frac{A(t_k) - \lambda t_k}{\alpha\sqrt{t_k}} < \frac{k - \lambda t_k}{\alpha\sqrt{t_k}}\right).$$

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 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. □

Hints

h.1.19.1 Take $c = 1$.

h.1.19.7 Let $S_n = \sum_{k=1}^n X_k$ and $\sigma_n^2 = n\,\mathsf{V}[X]$. Then, under conditions you can find on the internet,

$$\frac{S_n - n\,\mathsf{E}[X]}{\sigma_n} \to \mathcal{N}(0,1), \quad \text{as } n \to \infty,$$

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

$$\frac{S_n - n\,\mathsf{E}[X]}{\sigma_n} \approx \mathcal{N}(0,1) \iff$$

$$
\begin{aligned}
S_n - n\,\mathsf{E}[X] &\approx \sigma_n \mathcal{N}(0,1) \iff \\
S_n - n\,\mathsf{E}[X] &\approx \mathcal{N}(0,\sigma_n^2) \iff \\
S_n &\approx n\,\mathsf{E}[X] + \mathcal{N}(0,\sigma_n^2) \iff \\
S_n &\approx \mathcal{N}(n\,\mathsf{E}[X],\sigma_n^2),
\end{aligned}
$$

hence, S_n is approximately distributed as $\mathcal{N}(n\,\mathsf{E}[X],\sigma_n^2)$.

h.1.19.8 It took me some time to understand the details of part a. I then decided to read part b first, which clarified the intent of part a.

Solutions

s.1.19.1 If $c = 1$, as is the case for the $M/M/1$ queue, $\sqrt{2(c+1)} - 1 = 2 - 1 = 1$, so that (1.94) reduces to $\mathsf{E}\left[W_Q\right] = \rho/(1-\rho)\,\mathsf{E}[S]$. Recall that $C_a^2 = C_s^2 = 1$ for the $M/M/1$.

s.1.19.2 Not necessarily. Since jobs do not arrive in general as a Poisson process, we cannot use the PASTA property to conclude that the time average queue length $\mathsf{E}\left[L_Q\right]$ is the same as the average queue length as observed by customers.

(Can you provide an example that shows this difference? Hint, we did this earlier.)

s.1.19.3 This follows from the observation that $\lambda(1-\beta)$ is the net arrival rate, as jobs are lost at a rate $\lambda\beta$. Hence, the load must be $\lambda(1-\beta)/\mu$. As the load must be equal to ρ, it follows that $\rho = \lambda(1-\beta)/\mu$, from which the other result immediately follows.

s.1.19.4 $\rho = \lambda\,\mathsf{E}[S] = 1/60 \cdot 50 = 5/6$. Since job arrivals do not overlap any job service, the number of jobs in the system is 1 for 50 seconds, then the server is idle for 10 seconds, and so on. Thus $\mathsf{E}[L] = 1 \cdot 5/6 = 5/6$. There is no variance in the interarrival times, and also not in the service times, thus $C_a^2 = C_s^2 = 0$. Also $\mathsf{E}\left[W_Q\right] = 0$ since $\mathsf{E}\left[L_Q\right] = 0$. Interestingly, we get the same from Kingman's formula.

s.1.19.5 Again $\mathsf{E}[S]$ is 50 seconds, so that $\rho = 5/6$. Also $C_a^2 = 0$. For the C_S^2 we have to do some work.

$$
\begin{aligned}
\mathsf{E}[S] &= \frac{20}{2} + \frac{80}{2} = 50 \\
\mathsf{E}\left[S^2\right] &= \frac{400}{2} + \frac{6400}{2} = 3400 \\
\mathsf{V}[S] &= \mathsf{E}[S]^2 - (\mathsf{E}[S])^2 = 3400 - 2500 = 900 \\
C_S^2 &= \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} = \frac{900}{2500} = \frac{9}{25}.
\end{aligned}
$$

s.1.19.6 All follows straightaway from the definitions in the main text. In the $G/G/1$ queue jobs arrive and depart in single units. Hence, $|A(n,t)-D(n,t)| \leq 1$, c.f., Eq. (1.40a). Then,

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

The left hand side goes to $\lambda\pi(n)$ as $t \to \infty$, and the right hand side to $\delta\delta(n)$. Use the fact that we always assume, implicitly, that the system is stable, so that $\lambda = \delta$. As a consequence $\delta(n) = \pi(n)$.

s.1.19.7 If the number in the system is n, then the total service time required to process these n jobs is the sum of the service times, i.e.,$W_{Q,k} = \sum_{k=1}^{n} S_k$ where k counts over the n jobs in the system. Then, with the hint,

$$W_{Q,k} \sim \mathcal{N}(n\,\mathsf{E}[S], \sigma_n^2)$$

where $\sigma_n^2 = n\mathsf{V}[S]$. Thus, the waiting time in queue is approximately normally distributed with mean $n\,\mathsf{E}[S]$ and variance $n\mathsf{V}[S]$.

s.1.19.8 1. First the $G/G/1$ case.

```
>>> a = 3.
>>> b = 9.
>>> EA = (b+a)/2.
>>> EA
6.0
>>> labda = 1./EA # per minute
>>> labda
0.16666666666666666
>>> VA = (b-a)*(b-a)/12.
>>> CA2 = VA/(EA*EA)
>>> CA2
0.08333333333333333

>>> ES = 5.
>>> sigma = 2
>>> VS = sigma*sigma
>>> CS2 = VS/(ES*ES)
>>> CS2
0.16
```

```
>>> rho = labda*ES
>>> rho
0.8333333333333333

>>> Wq = (CA2+CS2)/2. * rho/(1.-rho) * ES
>>> Wq
3.041666666666665
```

2. Now the $M/G/1$ case.

```
>>> Wq = (1.+CS2)/2.*rho/(1.-rho)*ES
>>> Wq
14.499999999999993
```

The arrival process with uniform interarrival times is much more regular than a Poisson process. In the first case, bus arrivals are spaced in time at least with 3 minutes.

1.20 Setups and Batch Processing

Theory and Exercises

With the $G/G/1$ waiting time formula (1.94) we can compute, approximately, the waiting time in queue for many non-trivial queueing situations. In this section we focus on the effect of change-overs, or setups. Consider, for instance, a machine that paints red and blue bikes. When the machine requires a color change, a clean-up time is necessary. As we will see it is necessary in such situations to produce in batches. Other examples are ovens that need warm up or cool down times when different item types require different temperatures. In service settings, when servers have to move from part of a building to another, they time spend moving cannot be spent on serving customers.

Specifically, we analyze the following queueing situation. There are two job families, e.g., red and blue. Jobs arrive at rate λ_r and λ_b, respectively, so that the arrival rate of jobs is $\lambda = \lambda_b + \lambda_r$. For ease we assume that a job service time S_0 have the same distribution for both colors. The change-over time is given by a random variable T, which is independent of the normal job service times.

Jobs of each color are assembled into batches of size B. Once a batch is complete, the batch enters a queue (of batches). Once a batch reaches the head of the queue, the machine performs a setup, and then starts processing each job individually until the batch is complete. If there is another batch in queue, a new setup time is

required. Otherwise the machine just switches off. 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 finish.

We analyze in steps the total average time a job spends in the system.

First we consider the time it takes to form a batch.

1.20.1. Show that the total time to form a red batch is $(B-1)/\lambda_r$. Hence, the average time a red job spends waiting until the batch is complete is

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

Now that we know how long jobs spend to form batches, we turn to finding an estimate for the average time a batch has to spend in queue, for which we use Eq. (1.94). Recall that for this formula, we need the arrival rate, the average service time and the SCVs. These elements we will compute now.

It is evident that the rate at which batches arrive is

$$\lambda_B = \frac{\lambda}{B},$$

since both job colors have the same batch size.

Observe next that the machine not only serves jobs, part of the time it is occupied with setups. This leads to the idea to *incorporate* the effects of the setup times in the service times. For this we distinguish between a job's *net service time* S_0 and its *effective* processing time S_e which also include setup times.

1.20.2. Show that

$$\mathsf{E}[S_e] = \mathsf{E}[S_0] + \frac{\mathsf{E}[T]}{B}.$$

Now that the batch arrival rate and the service time per batch are known, the load can be written as

$$\rho = \lambda_B(B\,\mathsf{E}[S_0] + \mathsf{E}[T]) = \lambda\left(\mathsf{E}[S_0] + \frac{\mathsf{E}[T]}{B}\right),$$

where the first equality has the interpretation of the batch arrival rate times times the work per batch, while the second is the job arrival rate times the effective work per job.

1.20.3. Show that the requirement $\rho < 1$ leads to the following constraint on the minimal batch size B

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

The next element is to find the SCVs. To obtain $C^2_{a,B}$, i.e., the SCV of the interarrival times of the batches, recall that jobs are first assembled into batches, and then these batches are sent to the queue.

1.20.4. Show that

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

with C^2_a the SCV of interarrival times of individual jobs.

The last element is to find the SCV $C^2_{e,B}$ of the service times.

1.20.5. Show that

$$C^2_{e,B} = \frac{B\mathsf{V}[S_0] + \mathsf{V}[T]}{(B\mathsf{E}[S_0] + \mathsf{E}[T])^2}.$$

Finally, when the batch is taken into service, there can be various rules to determine when the job's service finished. If the job has to wait until all jobs in the batch are served, the time a job spends at the server is $B\mathsf{E}[S_0] + \mathsf{E}[T]$.

1.20.6. In our model we assume that jobs can leave right after being served. Show for this case that the expected time until a job leaves the server is

$$\mathsf{E}[T] + \frac{B-1}{2}\mathsf{E}[S_0] + \mathsf{E}[S_0].$$

Clearly, we now have all elements to compute the average time in the system. Lets illustrate this.

1.20.7. Jobs arrive at $\lambda = 3$ per hour at a machine with $C^2_a = 1$; service times are exponential with average 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. Suppose the batch size $B = 30$ jobs. What is the minimal

batch size? What is the average time, a red job spends in the system?

In summary, to find the average queueing time in the system, we need to find the arrival rate, the effective service times and the SCVs, so that we can fill in the $G/G/1$ waiting time formula. The main idea is to incorporate the setup times into the job service times. Observe also that the times to form and process batches are linear functions of the batch size B, while the load is, for small batch sizes, very sensitive to the batch size. Thus, batch sizes should not be too small. Overall, batch sizes need to be tuned to minimize the total average xtime jobs spend in the system. When the batch sizes are small, the load ρ is near to one (in other words, the server spends a relatively large fraction of its time on setups), so that the queueing times are long, but the times to form a batch are small. If, however, the batch sizes are large, the queueing times will be relatively short, but the times to form and unpack batches will be large.

Hints

h.1.20.2 What fraction of the setup time $\mathsf{E}[T]$ 'belongs' to one job?

h.1.20.5 What is the variance of a batch service time?

Solutions

s.1.20.1 Suppose a batch is just finished. The first job of a new batch needs to wait, on average, $B-1$ interarrival times until the batch is complete, the second $B-2$ interarrival times, and so on. The last job does not have to wait at all. Thus, the total time to form a batch is $(B-1)/\lambda$.

An arbitrary job can be anywhere in the batch, hence the average time a job must wait until the batch is complete is half the total time.

s.1.20.2 The total service time spent on a batch of size B is $B\,\mathsf{E}[S_0]+\mathsf{E}[T]$. The effective time per job is then the average, i.e., $(B\,\mathsf{E}[S_0]+\mathsf{E}[T])/B$.

s.1.20.4 The variance of the interarrival time of batches is B times the variance of job interarrival times. The interarrival times of batches is also B times the interarrival times of jobs. Thus,

$$C^2_{a,B} = \frac{B\mathsf{V}[X]}{(B\,\mathsf{E}[X])^2} = \frac{\mathsf{V}[X]}{(\mathsf{E}[X])^2}\frac{1}{B} = \frac{C_a^2}{B}.$$

s.1.20.5 The variance of a batch is $\mathbb{V}\left[\sum_{i=1}^{B} S_{0,i} + T\right] = B\,\mathbb{V}[S_0] + \mathbb{V}[T]$, since the normal service times $S_{0,i}, i = 1,\ldots,B$ of the job are independent, and also independent of the setup time T of the batch.

s.1.20.6 First, wait until the setup is finished, then wait (on average) for half of the batch (minus the job itself) to be served, and then the job has to be served itself.

s.1.20.7 First check the load.

```
>>> labda = 3 # per hour
>>> ES0 = 15./60 # hour
>>> ES0
0.25
>>> ET = 2.
>>> B = 30
>>> ESe = ES0+ ET/B
>>> ESe
0.31666666666666665

>>> rho = labda*ESe
>>> rho
0.95
```

Evidently, the load is smaller than 1.

The minimal batch size is

```
>>> Bmin = labda*ET/(1-labda*ES0)
>>> Bmin
24.0
```

So, with $B = 30$ we are on the safe side.

The time to form a red batch is

```
>>> labda_r = 0.5
>>> EWf = (B-1)/(2*labda_r)
>>> EWf # in hours
29.0
```

Now the time a batch spends in queue

```
>>> Cae = 1.
>>> CaB = Cae/B
```

```
>>> CaB
0.03333333333333333
>>> Ce = 1 # SCV of service times
>>> VS0 = Ce*ES0*ES0
>>> VS0
0.0625
>>> VT = 1*1. # Var setups is sigma squared
>>> VSe = B*VS0 + VT
>>> VSe
2.875
>>> ESB = B*ES0+ET
>>> CeB = VSe/(ESB*ESB)
>>> CeB
0.03185595567867036
>>> ESb = B*ES0 + ET
>>> ESb
9.5
>>> EWq = (CaB+CeB)/2 * rho/(1-rho) * ESb
>>> EWq
5.8833333333333275
```

The time to unpack the batch, i.e., the time at the server.

```
>>> ES = ET + (B-1)/2 * ES0 + ES0
>>> ES
5.875
```

The overall time red jobs spend in the system.

```
>>> total = EWf + EWq + ES
>>> total
40.758333333333326
```

1.21 Non-preemptive Interruptions, Server Adjustments

Theory and Exercises

Besides setups, other types interruptions can occur, for example, small adjustments required by a machine after serving a few jobs. Interruptions, or outages, occuring *between* two jobs are called *non-preemptive* as they do not preempt the processing of

a job. In this section we develop a simple model to incorporate the effects of such non-preemptive outages in the service times of jobs, so that we can use the $G/G/1$ waiting time formula.

1.21.1. What makes outages different from service time lost due to setup times between batches, as discussed in Section 1.20?

Let us assume that adjustments occur geometrically distributed, i.e., have the memoryless property in discrete time, between any two jobs with a mean of B jobs between any two repairs. Thus, the probability of a repair between two jobs is $p = 1/B$.

1.21.2. Show that the average *effective* processing time S_e, i.e., the time the service is occupied with processing jobs including adjustments, is

$$\mathsf{E}[S_e] = \mathsf{E}[S_0] + \frac{\mathsf{E}[T]}{B}.$$

Thus the effective server load including downtimes is $\rho = \lambda \mathsf{E}[S_e]$.

1.21.3. Show that

$$\mathsf{E}\left[S_e^2\right] = \mathsf{E}\left[S_0^2\right] + 2\frac{\mathsf{E}[S_0]\,\mathsf{E}[T]}{B} + \frac{\mathsf{E}\left[T^2\right]}{B}.$$

1.21.4. Use the above to find that

$$\mathsf{V}[S_e] = \mathsf{V}[S_0] + \frac{\mathsf{V}[T]}{B} + (B-1)\left(\frac{\mathsf{E}[T]}{B}\right)^2.$$

With the above we can compute $c_e^2 = \mathsf{V}[S_e]/(\mathsf{E}[S_e]^2)$ of the effective job processing times. We have now all elements to fill in the $G/G/1$ waiting time formula!

1.21.5. A machine requires an adjustment with average 5 hours and 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 hour. Interruptions occur on average after 30 jobs. Compute the average waiting time in queue.

Observe that with these formulas we can obtain quantitative insights into the effects of reducing adjustment times, or the variability of these adjustments times. In particular, sensitivity analysis is very relevant for practical cases to find out the dominating parameters.

Hints

h.1.21.2 If there is no failure, the service time is S_0. If there is a failure, the service of a job is $T + S_0$, since we add the outage time to the service time of the job.

h.1.21.5 Get the units right. First compute the load, and then compute the rest.

Solutions

s.1.21.1 Setups are planned, in the model of Section 1.20, between B jobs. Thus, the setups do not occur between any two jobs. Random interruptions, on the other hand, can potentially happen anywhere and between any two jobs.

s.1.21.2

$$\mathsf{E}[S_e] = (1-p)\mathsf{E}[S_0] + p(\mathsf{E}[T] + \mathsf{E}[S_0]) = \mathsf{E}[S_0]\frac{B-1}{B} + (\mathsf{E}[S_0] + \mathsf{E}[T])\frac{1}{B},$$

since $p = 1/B$.

s.1.21.3

$$\begin{aligned}\mathsf{E}\left[S_e^2\right] &= \mathsf{E}\left[S_0^2\right]\frac{B-1}{B} + \mathsf{E}\left[(S_0+T)^2\right]\frac{1}{B} \\ &= \mathsf{E}\left[S_0^2\right]\frac{B-1}{B} + \mathsf{E}\left[S_0^2\right]\frac{1}{B} + 2\mathsf{E}[S_0]\,\mathsf{E}[T]\frac{1}{B} + \mathsf{E}\left[T^2\right]\frac{1}{B}\end{aligned}$$

s.1.21.4

$$\begin{aligned}\mathsf{V}[S_e] &= \mathsf{E}\left[S_e^2\right] - (ES_e)^2 \\ &= \mathsf{E}\left[S_0^2\right] + 2\mathsf{E}[S_0]\,\mathsf{E}[T]\frac{1}{B} + \mathsf{E}\left[T^2\right]\frac{1}{B} \\ &\quad -(\mathsf{E}[S_0])^2 - 2\mathsf{E}[S_0]\,\mathsf{E}[T]\frac{1}{B} - (\mathsf{E}[T])^2\frac{1}{B^2} \\ &= \mathsf{V}[S_0] + ((\mathsf{E}\left[T^2\right] - (\mathsf{E}[T])^2)\frac{1}{B} + (\mathsf{E}[T])^2\left(\frac{1}{B} - \frac{1}{B^2}\right).\end{aligned}$$

s.1.21.5 First we determine the load.

```
>>> B=30
>>> ES0 = 1.5
>>> labda = 9/(2*8) # arrival rate per hour
>>> ET=5
>>> ESe=ES0+ET/B
>>> ESe
1.6666666666666667
>>> rho = labda*ESe
>>> rho
0.9375
```

So, at least the system is stable.

```
>>> VS0 = 0.5*0.5
>>> VT = 2*2
>>> VSe = VS0 + VT/B + (B-1)*(ET/B)**2
>>> VSe
1.1888888888888887
>>> Ce2 = VSe/(ESe*ESe)
>>> Ce2
0.4279999999999999
```

And now we can fill in the waiting time formula

```
>>> Ca2=1 # Poisson arrivals
>>> EW = (Ca2+Ce2)/2 * rho/(1-rho) * ESe
>>> EW
17.849999999999998
```

1.22 Preemptive Interruptions, Server Failures

Theory and Exercises

Up to now we assumed that servers are never interrupted while serving a job. In many situations this is not true, a person might be receive short phone calls 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 account for such *preemptive interruptions*, i.e., interruptions that occur *during* a service, and compute the influence on the mean waiting time.

Let us assume that a job's normal service, without interruptions, is given by S_0. The durations of the interruptions are given by the i.i.d. random variables $\{D_i\}$ and have common mean $\mathsf{E}[D]$ and variance $\mathsf{V}[D]$. If N interruptions occur, the effective service time will then be

$$S = S_0 + \sum_{n=1}^{N} D_i.$$

To compute the mean waiting time we can use the $G/G/1$ waiting time formula. From this formula it is clear that it suffices to find expressions for $\mathsf{E}[S]$ and $\mathsf{V}[S]$, since $C_e^2 = \mathsf{V}[S]/(\mathsf{E}[S])^2$. 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.

We first aim to find an expression for $\mathsf{E}[S]$. Write $S_N = \sum_{i=1}^{N} D_i$ for the total duration of the interruptions, so that the total job duration becomes $S = S_0 + S_N$.

1.22.1. Suppose that $N = n$, show that $\mathsf{E}[S_n] = n\,\mathsf{E}[D]$.

Let $p_n = \mathsf{P}(N = n)$; then it is reasonable that $\mathsf{E}[S_N] = \sum_{n=0}^{\infty} \mathsf{E}[S_n] p_n$. (Compare the definition of $\mathsf{E}[f(X)] = \sum_n f(n) p_n$.)

1.22.2. Use this expression to show that $\mathsf{E}[S_N] = \mathsf{E}[D]\,\mathsf{E}[N]$.

With the above,

$$\mathsf{E}[S] = \mathsf{E}[S_0 + S_N] = \mathsf{E}[S_0] + \mathsf{E}[D]\,\mathsf{E}[N].$$

To make further progress, we need some additional assumptions. A common assumption is that the time between two interruptions is memoryless, hence the time between failures is exponentially distributed, with rate λ_f say. Consequently, the number of interruptions N that occur during the net service time S_0 must be 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 to time to repair.

1.22.3. Show that

$$A = \frac{1}{1 + \lambda_f \mathsf{E}[D]}$$

for our model of interruptions.

1.22.4. Show that

$$\mathsf{E}[S] = \frac{\mathsf{E}[S_0]}{A} = \mathsf{E}[S_0](1 + \lambda_f \mathsf{E}[D]).$$

An intuitive way to obtain this result is by noting that A is the fraction of time the server is working. As the total service time of a job is $\mathsf{E}[S]$, the net work done is $A\,\mathsf{E}[S]$. But this must be the time needed to do the real job, hence $A\,\mathsf{E}[S] = \mathsf{E}[S_0]$.

It is important to realize that

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

hence the load increases due to failures.

We can similar ideas to derive an expression for the variance of S. The next exercise helps to understand why this derivation is a bit more involved.

1.22.5. Why is $\mathsf{V}[S] \neq \mathsf{V}[S_0] + \mathsf{V}\left[\sum_{i=0}^{N} D_i\right]$?

So let us first consider $\mathsf{E}\left[S^2\right]$; recall that $\mathsf{V}[S] = \mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2$, and we already know that $\mathsf{E}[S] = \lambda_f \mathsf{E}[S_0]$.

1.22.6. Show that

$$\mathsf{E}\left[S^2\right] = \mathsf{E}\left[S_0^2\right] + 2\mathsf{E}\left[S_0 \sum_{i=1}^{N} D_i\right] + \mathsf{E}\left[\sum_{i=1}^{N} D_i^2\right] + \mathsf{E}\left[\sum_{i=1}^{N} \sum_{j \neq i} D_i D_j\right]$$

For the last three components, we assume at first that S_0 is known, so that $N \sim P(\lambda_f S_0)$, hence $\mathsf{E}[N \mid S_0] = \lambda_f S_0$ and $\mathsf{E}\left[N^2 \,\middle|\, S_0\right] = \lambda_f S_0 + \lambda_f^2 S_0^2$.

1.22.7. Show that $\mathsf{E}\left[S_0 \sum_{i=1}^{N} D_i \,\middle|\, S_0\right] = \lambda_f S_0^2 \mathsf{E}[D]$.

1.22.8. Show that $\mathsf{E}\left[\sum_{i=1}^{N} D_i^2 \,\middle|\, S_0\right] = \lambda_f S_0 \mathsf{E}\left[D^2\right]$.

1.22.9. Show that $\mathsf{E}\left[\sum_{i=1}^{N} \sum_{j \neq i} D_i D_j \,\middle|\, S_0\right] = \lambda_f^2 S_0^2 (\mathsf{E}[D])^2$.

1.22.10. Combine the above to see that $\mathsf{E}\left[S^2 \,\middle|\, S_0\right] = \frac{S_0^2}{A^2} + \lambda_f \mathsf{E}\left[D^2\right] S_0$. From this,

$$\mathsf{E}\left[S^2\right] = \frac{\mathsf{E}\left[S_0^2\right]}{A^2} + \lambda_f \mathsf{E}\left[D^2\right] \mathsf{E}[S_0],$$

1.22.11. Next, show that

$$\mathsf{V}[S] = \frac{\mathsf{V}[S_0]}{A^2} + \lambda_f \mathsf{E}\left[D^2\right] \mathsf{E}[S_0].$$

1.22.12. Finally, show that

$$C_e^2 = \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} = C_0^2 + \frac{\lambda \mathsf{E}\left[D^2\right] A^2}{\mathsf{E}[S_0]}.$$

where C_0^2 is the SCV of S_0, i.e., the service time without interruptions.

If we assume that repair times are exponentially distributed with mean $\mathsf{E}[D]$, we can simplify this yet further.

1.22.13. Under this condition, show that $\mathsf{E}\left[D^2\right] = 2(\mathsf{E}[D])^2$.

1.22.14. With the above assumption on the distribution of D, show that

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

Again, we have all elements ready to use the $G/G/1$ waiting time formula. Lets illustrate this.

1.22.15. 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?

1.22.16. Suppose we could buy another machine that never fails. What is the average sojourn time?

Hints

h.1.22.1 Is it relevant that for the expection of S_n that $D_1, \ldots, D_n$ are independent?

h.1.22.3 Observe that $m_f = 1/\lambda_f$ and $m_r = \mathsf{E}[D]$?

h.1.22.4 Realize that $\mathsf{E}[N] = \lambda_f \mathsf{E}[S]_0$.

h.1.22.15 Mind to work in a consistent set of units, e.g., hours. Its easy to make mistakes.

Solutions

s.1.22.1 The expectation of the sum of random variables is the same as the sum of the expectations; independence is irrevelant. Hence,

$$\mathsf{E}[S_n] = \mathsf{E}\left[\sum_{i=1}^{n} D_i\right] = \mathsf{E}[D_1] + \mathsf{E}[D_2] + \cdots \mathsf{E}[D_n] = n\,\mathsf{E}[D],$$

where the last equation follows from the fact that the D_i have the same distribution.

s.1.22.2 Since $\mathsf{E}[S_n] = n\,\mathsf{E}[D]$,

$$\mathsf{E}\left[\sum_{i=1}^{N} D_i\right] = \sum_{n=0}^{\infty} p_n \mathsf{E}[S_n] = \sum_{n=0}^{\infty} p_n n\,\mathsf{E}[D] = \mathsf{E}[D]\,\mathsf{E}[N].$$

This result in known as Wald's equation.

s.1.22.3 The time to fail is the time in between to interruptions. We assumed that these times were exponential with mean $1/\lambda_f$. The duration of an interruption is D, which can be interpreted as the time to repair the server, hence $m_r = \mathsf{E}[D]$. With this

$$A = \frac{m_f}{m_r + m_f} = \frac{1/\lambda_f}{1/\lambda_f + \mathsf{E}[D]}.$$

s.1.22.4

$$\mathsf{E}[S] = \mathsf{E}[S_0] + \mathsf{E}[N]\,\mathsf{E}[D] = \mathsf{E}[S_0] + \mathsf{E}[D]\lambda_f\,\mathsf{E}[S_0] = \mathsf{E}[S_0](1 + \lambda_f\,\mathsf{E}[D]).$$

s.1.22.5 Because S_0 and N are not independent, given S_0, N_0 is Poisson distributed with mean $\lambda_f S_0$.

s.1.22.6 Just work out the square of $S_0 + \sum_{i=1}^{N}$ and take the expectations. Realize that $(\sum_i D_i)^2 = \sum_i D_i^2 + \sum_i \sum_{j \neq i} D_i D_j$.

s.1.22.7 $\mathsf{E}\left[S_0 \sum_{i=1}^{N} D_i \,\middle|\, S_0\right] = S_0\,\mathsf{E}\left[\sum_{i=1}^{N} D_i \,\middle|\, S_0\right] = S_0\,\mathsf{E}[D]\,\mathsf{E}[N] = \lambda_f\,\mathsf{E}[D] S_0^2$.

s.1.22.9 Since the $\{D_i\}$ are i.i.d,

$$\mathsf{E}\left[\sum_{i=1}^{N}\sum_{j\neq i} D_i D_j \,\middle|\, S_0\right] = \mathsf{E}[N(N-1)|S_0](\mathsf{E}[D])^2 = (\mathsf{E}\left[N^2|S_0\right] - \mathsf{E}[N\,|\,S_0])(\mathsf{E}[D])^2.$$

Now $\mathsf{E}\left[N^2 \,\middle|\, S_0\right] = \lambda_f^2 S_0^2 + \lambda_f S_0$ and $\mathsf{E}[N\,|\,S_0] = \lambda_f S_0$.

s.1.22.10 For the first equation,

$$\mathsf{E}\left[S^2 \,\middle|\, S_0\right] = S_0^2 + 2\lambda_f\,\mathsf{E}[D] S_0^2 + \lambda_f\,\mathsf{E}\left[D^2\right] S_0 + \lambda_f^2 (\mathsf{E}[D])^2 S_0^2.$$

Assemble all terms with S_0^2 and observe that $(1/A) = 1 + \lambda_f\,\mathsf{E}[D]$. For the second, recall that we assumed at first that S_0 was fixed, which we indicated by the condition on S_0. When S_0 is a random variable, we can just take the expectation at the left and right, and obtain the second result.

s.1.22.11

$$\mathsf{V}[S] = \mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2 = \frac{\mathsf{E}\left[S_0^2\right]}{A^2} + \lambda_f\,\mathsf{E}\left[D^2\right]\mathsf{E}[S_0] - \frac{(\mathsf{E}[S_0])^2}{A^2}.$$

s.1.22.12 Just realize that $\mathsf{E}[S] = \mathsf{E}[S_0]/A$, and use the above.

s.1.22.14 Since $A = 1/(1+\lambda_f \mathsf{E}[D])$,

$$\begin{aligned}\lambda \mathsf{E}\left[D^2\right] A^2 &= 2\lambda(\mathsf{E}[D])^2 A^2 = 2\lambda \mathsf{E}[D] A A \mathsf{E}[D] \\ &= 2\frac{\lambda \mathsf{E}[D]}{1+\lambda \mathsf{E}[D]} A \mathsf{E}[D] \\ &= 2\left(1 - \frac{1}{1+\lambda \mathsf{E}[D]}\right) A \mathsf{E}[D] = 2(1-A) A \mathsf{E}[D].\end{aligned}$$

s.1.22.15 Lets first compute the load. If $\rho > 1$ we are in trouble.

```
>>> labda = 4.
>>> ES0 = 10./60 # in hours
>>> labda_f = 1./3
>>> ED = 30./60 # in hours
>>> A = 1./(1+labda_f*ED)
>>> A
0.8571428571428571
>>> ES = ES0/A
>>> ES
0.1944444444444445
>>> rho = labda*ES
>>> rho
0.7777777777777778
```

As $\rho < 1$, the system is not in overload. Now for the queueing time.

```
>>> Ca2 = 1.
>>> C02 = 0. # deterministic service times
>>> Ce2 = C02 + 2*A*(1-A)*ED/ES0
>>> Ce2
0.7346938775510207
>>> EW = (Ca2+Ce2)/2 * rho/(1-rho) * ES
>>> EW
0.5902777777777779
>>> EW + ES # sojourn time
0.7847222222222223
```

s.1.22.16 Now we don't need to take availability into account: the machine never fails so $A = 1$.

```
>>> labda = 4.
>>> ES0 = 10./60 # in hours
>>> rho = labda*ES0
>>> rho
0.6666666666666666
>>> Ca2 = 1.
>>> C02 = 0. # deterministic service times
>>> EW = (Ca2+C02)/2 * rho/(1.-rho) * ES
>>> EW
0.19444444444444442
>>> EW + ES # sojourn time
0.38888888888888884
```

The average time in queue reduces for ≈ 0.6 to ≈ 0.2 hours, a reduction by a factor 3.

2 Open Queueing Networks

2.1 Deterministic Queueing Networks

Theory and Exercises

The simplest possible single-server queueing system is such that job interarrival times are deterministic and all service times are equal. In such cases it is easy to determine the maximal throughput, i.e., the capacity, of the queueing system. Clearly, if the service time is t_1, then the service rate is $r_1 = 1/t_1$. We can also determine the minimal amount of jobs required to keep the server busy, hence achieve maximal throughput. We call this the *critical work in progress (WIP)* and denote it by W_0. The minimal amount of time needed to move from the start of the system to the end, also called *raw processing time*, is denoted by T_0. With Little's law we can relate these two concepts for the single-server example. If the arrival rate is equal to the service rate, then $\lambda = r_1$, so that

$$W_0 = \lambda T_0 = r_1 T_0.$$

Since, in the single server case $T_0 = t_1$, we see that $W_0 = r_1 t_1 = t_1^{-1} t_1 = 1$. This result is evident: to get maximal throughput, it is necessary that the server is always busy, hence it must be that $W_0 \geq 1$. Given that we deal with a $D/D/1$ queue, by assumption, we can tune the arrivals such that whenever a job leaves, a new job arrives.

When we deal with *networks*, determining W_0 and T_0 is not so simple anymore. These quantities, however, are important to know. Clearly, jobs need at least the raw processing time T_0 in the network, which is the sum of the required processing times but does not include, by assumption, any queueing times. The critical WIP W_0 is also important, because if the network contains less jobs than this amount, throughput will suffer. The reason is that when there are not sufficient jobs in the network, the bottleneck machines are not always fully loaded with work. For these reasons T_0 and W_0 act as lower bounds on the time jobs spend in the network and the amount of work required to achieve a certain throughput. Finally, it is important to know the bottleneck capacity r_b of the network as the maximal throughput of the network cannot exceed the bottleneck capacity. Thus, when accepting jobs, one needs to respect the capacity of the network.

Before we study queueing networks in stochastic settings, we should familiarize ourselves with the behavior of networks of queueing systems in the simplest setting possible. For this we need some further definitions. A queueing network can be represented as *graph*. *Stations* form the nodes in the graph, and edges between nodes represents the routing, which is the possibility that jobs can move from one station to another. A simple example is a tandem network with two stations. Jobs arrive at station 1 and after service they move to station 2. After service at station 2, they leave the network. Also, a station can contain one or more servers. For instance, a single-server station contains just one server. Finally, we assume that job service times are constant at all servers at all stations. In other words, service times need not be same from one machine/server or station to another, but each server works at constant speed. The *utilization* of station i is the rate $\lambda_i t_i$, i.e., the rate at which work arrives, times the (average) service time per job. The *bottleneck* station is the station with the *highest utilization*.

As an example, consider n single-server stations in tandem such that all stations have the same service time S. The raw processing time is then

$$T_0 = nS. \tag{2.1}$$

2.1.1. Suppose we would release just one job in the above tandem system so that only when the first job has visited all stations, the next job is released. What is the throughput of the network?

2.1.2. Motivate that if we are prepared to release w jobs, the throughput of the network becomes

$$\mu(w) = \min\{w, n\}\frac{1}{T_0}.$$

Conclude that we need a minimal amount w in the network to guarantee that the network's throughput $\mu(w)$ exceeds the arrival rate λ, that is, there is a minimal w such that $\mu(w) \geq \lambda$.

2.1.3. Show that only when $w \geq n$.

$$\mu(w) = \frac{1}{S},$$

in which case the maximal capacity of the network is achieved.

Finally, for given w the largest arrival rate we can handle is $\lambda = \mu(w)$.

2.1.4. Show that when $\lambda = \mu(w)$, the waiting time in the network is

$$\mathsf{E}[W] = \begin{cases} T_0 = nS, & \text{if } w \leq n, \\ \frac{w}{n} T_0 = wS, & \text{if } w \geq n, \end{cases}$$

2.1.5. What do you conclude from the above exercises?

The problems below are generalizations to networks with multi-server stations, servers with different processing rates, and multiple routings. They are meant to help you become acquainted with networking behavior, and also with Little's law, which we we use time and again here[1].

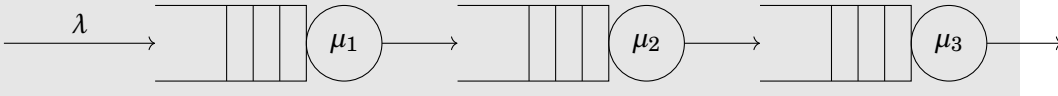
2.1.6. A production system consists of 2 separate stations. Jobs never go from one machine to another. The processing times are $(t_1, t_2) = (1, 2)$ hours. The arrival rates are $(\lambda_1, \lambda_2) = (1/3, 1/3)$ per hour. What is the fastest machine? Which machine is the bottleneck?

2.1.7. In the network of the previous exercise, what are T_0 and W_0?

2.1.8. A production network consists of 3 single-machine stations in tandem, i.e., in series, like the sketch below. The processing times are $(2, 3, 2)$ hours respectively, i.e., constant and such that $t_1 = 2$ hours, $t_2 = 3$ hours and $t_3 = 2$ hours. What are the critical WIP W_0, bottleneck capacity r_b and raw processing

[1] Warning, these problems are quite challenging, despite their apparent simplicity; I had much less trouble making the problems than solving them. In fact, to ensure that I got the answer right, I often tried to find at least two different ways to solve the same problem.

time T_0?



2.1.9. A production network consists of 3 single-machine stations in tandem with processing times $(2,3,2)$ hours. Suppose we modify this network by adding one extra machine to station 2, with processing time $t = 3$ hours. What are W_0, r_b and T_0?

2.1.10. A production network consists of 3 single-machine stations in tandem with processing times $(2,3,2)$ hours. Suppose we modify this network by adding one extra machine to station 2 with processing time $t = 4$ hours. What are W_0, r_b and T_0?

2.1.11. Why does the critical WIP change in the different cases above?

2.1.12. A production network consists of 3 single-machine stations in tandem with processing times $(2,3,2)$ hours. Suppose we modify this network by adding one extra machine to station 2 with processing time $t = 1$ hour. What are W_0, r_b and T_0?

2.1.13. In the previous question, it might seem that one machine at the second station is superfluous. Which one, and why? Would you remove this superfluous machine in a real production network?

2.1.14. A production network consists of 3 single-machine stations in tandem with processing times $(2,3,2)$ hours. Suppose we modify this network by adding one extra machine to station 2 with processing time $t = 7$ hours. What are W_0, r_b and T_0?

2.1.15. A production network consists of 3 single-machine stations in tandem with processing times $(2,3,2)$ hours. Suppose we modify this network such that 2/3 of the jobs *leave after the first* station, i.e., between station 1 and 2. What are W_0, r_b and T_0?

2.1.16. A production network consists of 3 single-machine stations in tandem with processing times $(2,3,2)$ hours. Suppose we modify this network such that 4/5 of the jobs *bypass the second* station, i.e., 1/5 of the jobs have the routing $[1,2,3]$, but 4/5 have routing $[1,3]$. What are W_0, r_b and T_0?

Production situations can be pretty complicated. Jobs can enter the network at different places, not just at station 1 as we considered here. Jobs may also need rework at one or more stations. One consequence of these aspects is that the slowest machines or stations need not be the bottlenecks. If you like, you can try to find or develop a general algorithm to compute T_0, W_0 and r_b for any network of reasonable size. One way to approach this problem might be to place (imaginary) huge amounts of work in front of each station and just see how the network drains. Given the elegance of the problem, I am quite sure it has been solved, but I haven't as yet.

Hints

h.2.1.4 Use Little's law.

h.2.1.5 If $w < n$, then what happens? If $w > n$, what are the $w - n$ jobs doing?

h.2.1.9 Is the new machine placed in parallel or in series?

h.2.1.10 Check your reasoning very carefully here. It is easy to do it wrong.

Solutions

s.2.1.1 In this case, the throughput μ of the network is $1/T_0$: only one server at a time is serving a job, the rest of the servers are forced to be idle as we only allow one job at a time in the network.

s.2.1.2 If there is one job in the system, $\mu(1) = 1/T_0$. If there are two, both jobs can be simultaneously worked on, each at a different station, hence $\mu(2) = 2/T_0$. And so on, until all stations are filled, at the same time, by a job, in which case $\mu(n) = n/T_0$. When there are yet more jobs in the network, only n can be simultaneously in service, the rest must wait at some station.

s.2.1.3 Using (2.1),

$$\mu(w) = \frac{n}{T_0} = \frac{n}{nS} = \frac{1}{S}.$$

s.2.1.4

$$\mathsf{E}[W] = \frac{\mathsf{E}[L]}{\lambda} = \frac{w}{\mu(w)} = \frac{w}{\min\{w,n\}} T_0 = \begin{cases} T_0 = nS, & \text{if } w \le n, \\ \frac{w}{n} T_0 = wS, & \text{if } w \ge n. \end{cases}$$

s.2.1.5 When $w > n$, there must be $w - n$ jobs in queue and the other n jobs are in service. So, in conclusion, in deterministic tandem networks adding more than n jobs does not increase the network's throughput, but only increases the waiting times. On the other hand, setting $w < n$ minimizes the waiting times, but also negatively affects the throughput of the network.

Thus, to balance waiting times and throughput, we need to tune the amount of work in the system. The critical WIP W_0 is the minimal WIP we need to keep all bottleneck machines occupied without adding extra waiting time.

s.2.1.6 The processing rates are $\mu_1 = 1/1$ and $\mu_2 = 1/2$ per hour, respectively. The utilizations $\rho_i = \lambda_i/\mu_i$ are 1/3 and 2/3, respectively. Thus, machine 2 is the bottleneck: it has the highest utilization.

s.2.1.7 Since these machines are not related by a routing (jobs going from one machine to another), computing T_0 and so on does not make much sense for the network. We can just consider each machine on its own. As we need 2 jobs to keep each machine busy, it follows that $W_0 = 2$.

We can use Little's law to find T_0. The rate at which jobs can be served is $\mu = 1 + 1/2 = 3/2$. Thus,

$$T_0 = W_0/\mu = 2/(3/2) = 4/3.$$

Clearly, the average time a job spends in this system is 4/3 time units if jobs arrive at the rate they are served. (Recall, everything is deterministic.)

Can we see this in another way? Yes, because 2 out of 3 jobs have service time of 1, and 1 out of 3 a service time of 2,

$$\frac{2}{3}1 + \frac{1}{3}2 = \frac{4}{3}.$$

s.2.1.8 The raw processing time is just the sum of the processing times at each station. All stations have the same arrival rate, hence machine, being the slowest, is the bottleneck. Hence,

$$T_0 = 2+3+2 = 7$$
$$r_b = \mu_2 = \frac{1}{3}$$
$$W_0 = r_b T_0 = \frac{7}{3}.$$

s.2.1.9 Realize that machines at a station are in parallel, not in series. Thus, when we add capacity to station 2, we don't add an extra processing step to station 2, we increase capacity.

Now

$$r_2 = \frac{1}{3} + \frac{1}{3} = \frac{2}{3},$$

so that station 2 is no longer the bottleneck. Station 1 and station 3 are the bottlenecks.

$$r_b = \frac{1}{2}$$
$$T_0 = 2+3+2 = 7$$
$$W_0 = r_b T_0 = \frac{7}{2}.$$

Often, students think that the processing rate is the inverse of the processing time, i.e., $t = r^{-1}$. This is not the case for multi-server stations. Thus, realize that the raw processing time at station 2 is still 3 hours, not 3/2.

s.2.1.10 Now,

$$r_2 = \frac{1}{3} + \frac{1}{4} = \frac{7}{12} > \frac{1}{2},$$

hence, station 1 and 3 are bottlenecks.

However, t_2 cannot be 3 hours anymore: some jobs will spend 4 hours at station 2 rather than 3 hours. In fact, the computation of the raw processing time at station 2, t_2 say, is a bit tricky. Some students reason that both machines serve half of the jobs, so that $T_2 = (3+4)/2$ hours. This, however, is not quite realistic. Like this, you would load the slow machine all of the time, and the faster machine would be idle quite a bit of the time.

Rather than spreading the jobs evenly over the two machines, we can also assume that each machine takes in work in proportion to its processing rate. Then we reason

like this. Consider 12 hours. In those 12 hours, the slow machine processes 3 jobs with duration 4 and the fast machine processes 4 jobs with duration 3. Therefore:

$$t_2 = \frac{3}{7}4 + \frac{4}{7}3 = \frac{24}{7}.$$

Another way to get this is like this. We can 'stuff' station 2 with jobs. The most jobs that fit in simultaneously is 2, one job per machine. Then, with Little's law,

$$t_2 = \frac{W}{r_2} = \frac{2}{7/12} = \frac{24}{7},$$

i.e., the same as the other answer.

Now, finally,

$$\begin{aligned} r_b &= \frac{1}{2} \\ T_0 &= 2 + \frac{24}{7} + 2 = 4 + 3 + \frac{3}{7} = 7\frac{3}{7} \\ W_0 &= r_b T_0. \end{aligned}$$

Finally, and the most realistic, is that the fast machine is always working, and the slow machine covers the rest. In that case, suppose that the fast machine processes jobs at rate r_1 and the slow at rate r_2. Then, since jobs arrive with rate r_a, the fast machine processes a fraction of r_1/r_a of the jobs, and the slow machine processes the left overs, i.e., a fraction of $(r_a - r_1)/r_a$ of the jobs. The raw processing time then becomes

$$\begin{aligned} t_2 &= \frac{r_1}{r_a}t_1 + \frac{r_a - r_1}{r_a}t_2 \\ &= \frac{1/3}{1/2}3 + \frac{1/2 - 1/3}{1/2}4 \\ &= 2 + \frac{1/6}{1/2}4 \\ &= 2 + \frac{4}{3} = 3\frac{1}{3}, \end{aligned}$$

and then,

$$T_0 = 2 + t_2 + 2 = 7\frac{1}{3},$$

and this is a tiny bit less then the previous way of organizing things.

So, why are the above answers different? One reason is that the objective is not explicitly stated. It might be that, to minimize time in a multi-server station, it is

optimal to let a fast machine idle and wait until the next job comes in. If we were to assign this next job to a slow machine, the time in the system may be longer. In general, it might be that schedules that minimize T_0 (on average) are not work-conserving (i.e., such that when jobs are waiting, not all machines are idle.)

s.2.1.11 Because T_0 increases and/or the bottleneck rate increases. In both cases, we need more W_0 to fill the network and achieve that the throughput can be equal to r_b in the best case.

s.2.1.12 Now the slow machine at station 2 becomes superfluous. The fast machine at station 2 can cope with all jobs that arrive from station 1.

$$\begin{aligned} r_b &= \frac{1}{2} \\ T_0 &= 2+1+2=5 \\ W_0 &= r_b T_0. \end{aligned}$$

Thus, why would you assign part of the jobs to the slow machine at station 2? There is no queue, the fast machine can cope with all the jobs. Moreover, assigning any job to the slow machines just adds to the cycle time.

Taking $T_2 = (1+3)/2 = 2$ is plainly silly.

s.2.1.13 See above. I would not remove the slow machine. In real life there is variability: the fast machine might fail for instance. Spare capacity is useful. However, if it is very costly to keep the slow machine operational, I would scrap it.

s.2.1.14

$$r_2 = \frac{1}{3} + \frac{1}{7} = \frac{10}{21} < \frac{1}{2}$$

hence station 2 is still the bottleneck

$$\begin{aligned} T_2 &= \frac{3}{10}7 + \frac{7}{10}3 = \frac{42}{10} = \frac{21}{5}\text{hour} \\ T_0 &= 2 + \frac{21}{5} + 2 \\ r_b &= \frac{10}{21} \\ W_0 &= r_b T_0. \end{aligned}$$

Note that we add capacity, so that we can process more jobs per unit time, but the raw processing time increases!

s.2.1.15 Now there are two loops. One third of the jobs pass all three stations. Station 1 can process these job at rate

$$\frac{1}{3}r_1 = \frac{1}{3}\frac{1}{2} = \frac{1}{6}$$

per hour. Since this is smaller than r_2 and r_3, station 1 is the bottleneck rate for this loop. Thus, W_0 in this loop is $r_b T_0 = 1/6 \cdot 7 = 7/6$.

The other loop only contains station 1. The bottleneck rate in this loop is

$$r_1\frac{2}{3} = \frac{1}{2}\frac{2}{3} = \frac{1}{3}.$$

Thus, $W_0 = r_b T_0 = 1/3 \cdot 2 = 2/3$ for this loop.

Hence, the total WIP is $7/6 + 2/3 = 11/6$ and

$$T_0 = \frac{1}{3}7 + \frac{2}{3}2 = \frac{11}{3},$$

since 1/3 take the long loop with $T = 7$ and 2/3 the short loop with $T = 2$. Thus, T_0 is the weighted average raw processing time.

Another way to analyze this situation is like this. Station 1 is the bottleneck, because its utilization is 1, while

$$\rho_2 = \lambda_2 t_2 = \frac{1}{2}\frac{1}{3}3 = \frac{1}{2}$$
$$\rho_3 = \lambda_3 t_3 = \frac{1}{2}\frac{1}{3}2 = \frac{1}{3}.$$

Since station 1 is the bottleneck in this network, $r_b = 1/2$. Since $T_0 = 11/3$, which we obtain as the weighted average computation above, we must have that

$$W_0 = r_b T_0 = \frac{1}{2}\frac{11}{3} = \frac{11}{6},$$

which agrees with our earlier result.

s.2.1.16 Now stations 1 and 3 are bottlenecks with $r_b = 1/2$. For T_0 we have

$$T_0 = \frac{1}{5}7 + \frac{4}{5}4 = \frac{23}{5}.$$

Thus,

$$W_0 = r_b T_0 = \frac{1}{2}\frac{23}{5} = \frac{23}{10}.$$

2.2 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 seem 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.

2.2.1. (Linear algebra refresher) Can you find an example to show for two matrices A and B that $AB \neq BA$, hence $xA \neq Ax$.

2.2.2. (Linear algebra refresher, 2) Suppose the matrix A has an eigenvalue 0. What is the geometric meaning of this fact?

2.2.3. Zijm.Ex.2.2.1

2.2.4. Zijm.Ex.2.2.2

2.2.5. Zijm.Ex.2.2.3

2.2.6. Zijm.Ex.2.2.4

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

2.2.8. Zijm.Ex.2.2.6

2.2.9. Show that Zijm.Eq.2.13 and 2.14 can be written as

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

2.2.10. Zijm.Ex.2.2.7

2.2.11. Zijm.Ex.2.2.8

2.2.12. (Hall 5.22). At a large hotel, taxi cabs arrive at a rate of 15 per hour, and parties of riders arrive at the rate of 12 per hour. Whenever taxicabs are waiting, riders are served immediately upon arrival. Whenever riders are waiting, taxicabs are loaded immediately upon arrival. A maximum of three cabs can wait at a time (other cabs must go elsewhere).

1. Let p_{ij} be the steady-state probability of there being i parties of riders and j taxicabs waiting at the hotel. Write the state transition equation for the system.

2. Calculate the expected number of cabs waiting and the expected number of parties waiting.

3. Calculate the expected waiting time for cabs and the expected waiting 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?

Hints

h.2.2.1 Let

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

.

h.2.2.12 This is really a neat problem. Please spend serious time on it to solve before looking at the answer. It requires some ingenuity on your part. Try to adapt the ideas behind Figure 2.2 of Zijm to this case.

Solutions

s.2.2.1

$$AB = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \neq \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} = BA.$$

Take $x = (1,1)$, then $xA = (1,2)$. Now, taking $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, we get $Ax = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$. Recall, horizontal vectors are not vertical vectors. The horizontal ones are to the left of a matrix, and the vertical ones to the right.

s.2.2.2 Many students think that a matrix is just a bunch of numbers ordered in a grid. This is, in my opinion, the most unproductive way to think about matrices. A much more useful way is to see a matrix as an *operator*. For instance, take A to be a 3×3 matrix. Then it can be seen as *mapping* from $\mathbb{R}^3$ to $\mathbb{R}^3$; it takes a vector $x \in \mathbb{R}^3$ and changes x into a new vector $Ax \in \mathbb{R}^3$. Thus, a square matrix A typically changes the direction of a vector x.

With this idea, consider the simple example with

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Clearly, A has an eigenvalue 0. Now take $v = (x, y, z)'$, so that

$$Av = A \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}.$$

We see that A removes any information about the z direction from the vector v. (It projects v on the $x-y$ plane, and throws away the z component of v.) But then, for a given vector $w = (x, y, 0)$ in the $x-y$ plane, it is impossible to use A to retrieve the original vector $v = (x, y, z)$. Thus, A cannot have an inverse on all of $\mathbb{R}^3$.

So, hopefully, with this example, you can memorize that for any matrix A to have an inverse, it is essential that it has no zero eigenvalues. When the *operator* A (don't think of a matrix as a set of numbers) throws away part of the dimension of the space on which it operates (i.e., it has one or more eigenvalue(s) 0), it is impossible to retrieve the part of the space it throws away. Hence, its inverse cannot be used to get this part of the space back.

s.2.2.3 Because jobs cannot leave faster than they arrive.

s.2.2.4 Observe from Exercise 1.10.3 that the inter-departure times of the $M/M/1$ queue are also independent and identically exponentially distributed with rate λ. Since the arrival process at the second station is the departure process of the first station, it must be that arrival process at the second station is also Poisson with rate λ. Interestingly, from the perspective of the second station it is as if there is not first station.

s.2.2.5 The question is not well specified. We know from Burke's law that the arrival *process* at the second station is Poisson. If, however, we know that station 1 is empty, then it is unlikely that a job will arrive at station 2 in the very near future.

Note that only the steady-state distributions of the queue lengths are independent. Once you have information about the state of one of the queues, then certainly this is not in 'steady-state'.

s.2.2.6 Simple algebra. (I am not going to write it out here. If you are willing to provide me the answer in latex I'll include it.)

s.2.2.7 Linear algebra is quite useful here!

Observe that P_{ij} is the probability that a job, after completing its service at node i, moves to node j. Then $\sum_{j=1}^{M} P_{ij}$ is the probability that a job moves from node i to another node in the network, i.e., stays in the network, while P_{i0} is the probability

that a job departing from node i leaves the network, in other words, the job is finished. When $\sum_{j=1}^{M} P_{ij} < 1$, then more jobs enter node i from the network than that node i sends 'back' into the network. Conceptually, node i 'leaks jobs'.

Now, consider some node k such that $P_{ki} > 0$, then the probability that a job that starts at node k, moves to node i and then leaves the network is equal to $P_{ki}Pi0$. Thus, since $P_{ki} > 0$ and $P_{i0} > 0$, the probability that a job leaves the network from note k in two steps is positive. More specifically, $P_{k0}^2 = \sum_{j=0}^{M} P_{kj}P_{j0} \geq P_{ki}P_{i0} > 0$.

The irreducibility assumption implies that in at most M steps it is possible to reach, with positive probability, any node from any other node in the network. Thus, for any node j to any other node k there is a sequence of nodes $j_1, j_2, \ldots j_{M-1}$ such that $P_{jk}^M \geq P_{jj_1}P_{j_1j_2}\cdots P_{j_{M-1}k} > 0$.

Thus, if there is a node i such that $P_{i0} > 0$, then it is possible from any node that sends jobs to node i directly to leave the network in two steps. Likewise, when node i can be reached from node k in n steps, say, then $P_{k0}^{n+1} \geq P_{ki}^n P_{i0} > 0$, i.e., in at most $n+1$ steps it is possible to leave the network from such node k. This implies, in particular, that for all nodes $k = 1, 2, \ldots M$, i.e., all nodes in the network, $P_{k0}^{M+1} > 0$. For this reason we consider P^{M+1} in the hint.

As a final remark for students with knowledge of Markov chains, observe that the routing matrix P does not correspond to the transition matrix of a recurrent Markov chain. Since for at least one row i, $\sum_{j=1}^{N} P_{ij} < 1$, the matrix P is sub-stochastic. Hence, a Markov chain induced by P cannot be irreducible, because for this to happen, the chain must stay in some absorbing set with probability 1.

s.2.2.8 Since M is finite, and $k \leq M$, the set of numbers P_{k0}^{M+1} is finite. This, together with the fact that $P_{k0}^{M+1} > 0$ for all k, implies that there is some number $\epsilon > 0$ such that $P_{k0}^{M+1} > \epsilon$. Hence, for all entries $k = 1, 2, \ldots M$, we have that $P_{kj}^{M+1} < 1 - \epsilon$. This, in turn, implies that $P_{kj}^{2(M+1)} < (1-\epsilon)^2$, and so on, so that for any n, $P_{kj}^{n(M+1)} < (1-\epsilon)^n$. This implies, in more general terms, that the entries of P^n decrease at some geometric rate to 0.

It is well known that for any bounded sequence x_i and $0 \leq \alpha < 1$, $\sum_{i=0}^{\infty} x_i \alpha^i < \infty$. Applying this insight to the entries of P^n it follows that $\sum_{n=0}^{\infty} P_{jk}^n < \infty$.

Finally, applying $\lambda = \gamma + \lambda P$ recursively, we get

$$\lambda = \gamma + \lambda P = \gamma + (\gamma + \lambda P)P = \gamma(1+P) + \lambda P^2 = \gamma(1 + P + P^2) + \lambda P^3 \rightarrow \gamma \sum_{n=0}^{\infty} P^n.$$

By the above reasoning this last sum is well defined, and finite. (By the way, the above argument is not necessarily valid for matrices P that are infinite, since then $\inf\{P_{ik}^M\}$ need not be strictly positive.)

Another interesting way to see all this is by making the simplifying assumption that P is a diagonalizable matrix. (The argument can be generalized to include matrices reduced to Jordan normal form, but this gives optimal clutter, but does change the line of reasoning in any fundamental way.) In that case, there exists an invertible matrix V with the (left) eigenvectors of P as its rows and a diagonal matrix Λ with the eigenvalues on its diagonal such that

$$VP = \Lambda V.$$

Hence, premultiplying with V^{-1},

$$P = V^{-1}\Lambda V.$$

But then

$$P^2 = V^{-1}\Lambda V \cdot V^{-1}\Lambda V = V^{-1}\Lambda^2 V,$$

and in general $P^n = V^{-1}\Lambda^n V$. If each eigenvalue λ_i is such that its modulus $|\lambda_i| < 1$, then $\Lambda^n \to 0$ geometrically fast, hence $P^n \to n$ geometrically fast, hence the sequence of partial sums $\sum_{n=0}^{N} P^n$ must converge to some finite number as $N \to \infty$.

So, we are left with proving that the eigenvalues of P must have modulus less than 1. This fact follows from Gerschgorin's disk theorem, which I include for the interested student. Define the disk $B(a,r) = \{z \in \mathbb{C} || z - a| \leq r\}$, i.e., the set of complex numbers such that the distance to the center $a \in \mathbb{C}$ is less than or equal to the radius r. With this, the Gerschgorin disks of a matrix are defined as $B(a_{ii}, \sum_{j\neq i} |a_{ij}|)$, i.e., disks with center at the diagonal elements a_{ii} of A and and radius equal to the sum of the (modulus of the) elements of A on the ith row except a_{ii}. Then Gerschgorin's theorem says that all eigenvalues of A lie in the union of these disks, i.e., all eigenvalues $\lambda_i \in \bigcup_i B_{a_{ii}, \sum_{j\neq i} |a_{ij}|}$.

Assume for notational simplicity that for each row i of P we have that $\sum_j a_{ij} < 1$. (Otherwise apply the argument to P^{M+1}.) Then this implies for all i that

$$a_{ii} + \sum_{j\neq i} a_{ij} < 1.$$

Since all elements of P are non-negative, this also implies that

$$-1 < a_{ii} - \sum_{j\neq i} a_{ij} \leq a_{ii} + \sum_{j\neq i} a_{ij} < 1.$$

With this and using that a_{ii} is a real number (so that it lies on the real number axis) it follows that all elements in the disk $B(a_{ii}, \sum_{j\neq i} a_{ij})$ have modulus smaller than 1. As this applies to any row i, all disks lie strictly within the complex unit circle. But then, by Gerschgorin's theorem, all eigenvalues of P also lie strictly in the unit circle, hence all eigenvalues have modulus smaller than 1.

s.2.2.9 Take $n_i < c_i$. Then $\Pi_{k=1}^{n_i} \min\{k, c_i\} = \Pi_{k=1}^{n_i} k = n_i!$, and $(c_i\rho_i)^{n_i} = (\lambda_i/\mu_i)^{n_i}$. If $n_i \geq c_i$, then $\Pi_{k=1}^{n_i} \min\{k, c_i\} = c_i! c_i^{n_i - c_i}$, and $(c_i\rho_i)^{n_i} = (\lambda_i/\mu_i)^{n_i} c_i^{n_i}$..

s.2.2.10

$$P = \begin{pmatrix} \alpha & 1-\alpha \\ \beta_1 & \beta_2 \end{pmatrix}.$$

$$(\lambda_1, \lambda_2) = (\gamma, 0) + (\lambda_1, \lambda_2)P.$$

Solving first for λ_2 leads to $\lambda_2 = (1-\alpha)\lambda_1 + \beta_2\lambda_2$, so that

$$\lambda_2 = \frac{1-\alpha}{1-\beta_2}\lambda_1.$$

Next, using this and that $\lambda_1 = \alpha\lambda_1 + \beta_1\lambda_2 + \gamma$ gives with a bit of algebra

$$\begin{aligned} \gamma &= \lambda_1(1-\alpha) - \beta_1\lambda_2 \\ &= \lambda_1\left(1 - \alpha - \beta_1\frac{1-\alpha}{1-\beta_2}\right) \\ &= \lambda_1(1-\alpha)\left(1 - \frac{\beta_1}{1-\beta_2}\right) \\ &= \lambda_1(1-\alpha)\frac{1-\beta_1-\beta_2}{1-\beta_2}. \end{aligned}$$

Hence,

$$\lambda_1 = \frac{\gamma}{1-\alpha}\frac{1-\beta_2}{1-\beta_1-\beta_2}.$$

Thus,

$$\lambda_2 = \frac{1-\alpha}{1-\beta_2}\lambda_1 = \frac{\gamma}{1-\beta_1-\beta_2}.$$

We want of course that $\lambda_1 < \mu_1$ and $\lambda_2 < \mu_2$. With the above expressions this leads to conditions on α, β_1 and β_2. Note that we have three parameters, and two equations; there is not a single condition from which the stability can be guaranteed.

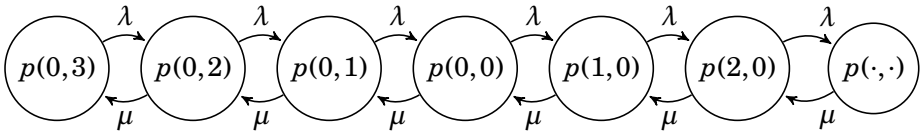
If $\alpha \uparrow 1$, the arrival rate at node 1 explodes. If $\beta_1 = 0$ no jobs are sent from node 2 to node 1.

s.2.2.11 Yes, the network remains a Jackson network. By Burke's law the departure process of each node is Poisson. In one of the earlier questions we derived that splitting (also known as thinning) and merging Poisson streams again lead to Poisson streams. The departures from node j to node k forms a thinned Poisson stream. The

external arrivals plus internal arrivals are merged into one Poisson stream, hence the arrivals at a station also form a Poisson stream.

Observe that the exponentiality of the service times and external inter-arrival times and Burke's law are essential for the argument.

s.2.2.12 1. Let p_{ij} be the fraction of time that the system contains i riders and j taxi cabs. I assume that all members of a party of riders can be served by a single cab (that is, the parties do not exceed the capacity of a cab and all members of a party have the same destination). For clarity, write μ for the rate at which cabs arrive, and λ for the arrival rate of parties of riders. Then the transitions are as in the figure below. Suppose first that there are 3 taxi cabs. When a group arrives (at rate λ), there is one taxi less, and so on, until there are no more taxi's left. Finally, if yet more groups arrive, they have to wait. When a new taxi arrives, the number of groups is reduced by one, and so on, until there are 3 taxi's waiting and no groups of people.



From this figure, we see that

$$\begin{aligned}
\lambda p_{0,3} &= \mu p_{0,2} \\
(\lambda+\mu)p_{0,2} &= \mu p_{0,1} + \lambda p_{0,3} \\
(\lambda+\mu)p_{0,1} &= \mu p_{0,0} + \lambda p_{0,2} \\
(\lambda+\mu)p_{0,0} &= \mu p_{1,0} + \lambda p_{0,1} \\
(\lambda+\mu)p_{1,0} &= \mu p_{2,0} + \lambda p_{0,0} \\
(\lambda+\mu)p_{2,0} &= \mu p_{3,0} + \lambda p_{1,0}
\end{aligned}$$

And so on. Thus, it is left to compute p_{ij}. Observe from this scheme, or the above figure, that the situation with the taxi's correspond to an $M/M/1$ queue, only the states have a 'different name'. Let q be the number of jobs in an M/M/1 queue. Some thought will reveal that the queueing system with cabs and parties can be mapped to an equivalent M/M/1 queueing system. In fact, consider the following table

j	i	q
3	0	0
2	0	1
1	0	2
0	0	3
0	1	4
0	2	5

and so on. Therefore, in general, it must be that

$$q = 3 - j + i.$$

From the M/M/1 queue we know right away that $p_q = \rho^q(1-\rho)$. With the above relation we can therefore immediately find that $p_{ij} = \rho^{3-j+i}(1-\rho)$, save that i and j must satisfy the constraints imposed by the model.

2. The expected number of cabs waiting must be

$$1p_{0,1} + 2p_{0,2} + 3p_{0,3}$$

and the expected number of parties waiting must be $\sum_{j=1}^{\infty} jp_{j,0}$

```
>>> labda = 12. # per hour
>>> mu = 15. # per hour
>>> rho = labda/mu

>>> def p(i,j):
...     q  = 3 - j + i
...     return rho**q*(1.-rho)
...
```

Expected number of cabs waiting:

```
>>> Lc = sum(j*p(0,j) for j in range(0,4))
>>> # Recall this sums up to 4, not including 4
>>> Lc
1.0479999999999998
```

To compute the expected number of parties waiting we formally have to sum to infinity. Rather than doing the algebra, I chose to truncate the summation at an i such that $\rho^i \ll 1$, i.e., negligble. Truncating at 30 seems reasonable enough:

```
>>> trunc = 30
>>> rho**trunc
0.0012379400392853823
```

Hm. At second thought this is not yet really small.

```
>>> trunc = 50
>>> rho**trunc
1.4272476927059638e-05
```

This is better. Now go for what we want to know:

```
>>> Lp = sum(i*p(i,0) for i in range(trunc))
>>> Lp
2.0476053945579213
```

3. This is tricky. I first, naively, computed $W_q = L_c/\mu$. This seems to make sense, as cabs arrive at rate μ, so that this expression follows from a standard application of Little's law. However, this is wrong, of course. When using Little's law to relate the number of jobs in queue (i.e., in the M/M/1 queue) and the queueing time we need to use λ, not μ. Similarly (and more formally by the mapping developed in part a), for our cab system we also need to use λ.

```
>>> Wq = Lc/labda
>>> Wq
0.08733333333333332
```

Thinking in templates is often useful, but makes one sloppy...

What would be the impact of allowing 4 cabs? Funny question, and with the above, trivial to answer.

```
>>> def p(i,j):
...     q  = 4 - j + i
...     return rho**q*(1.-rho)
...

>>> Lc = sum(j*p(0,j) for j in range(0,4))
>>> Lc
0.8383999999999999

>>> Lp = sum(i*p(i,0) for i in range(trunc))
>>> Lp
1.638084315646337
```

2.3 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.

2.3.1. 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

$$\mathsf{E}[W_Q] = \frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}, \tag{2.2}$$

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

2.3.2. Suppose we can remove all variability of service process at the second station. Show that in this case the total time in queue is equal to

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

2.3.3. Suppose now that we reduce the variability of the service process of the first station. Motivate that

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

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

2.3.4. 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 $C^2_{d,i}$ of the ith station i is the SCV of the arrival process $C^2_{a,i+1}$ at station $i+1$. Thus, if we have $C^2_{d,i}$ 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 $C^2_{d,i}$ 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 reasonable well approximated by the service process. If, however, the load is small, the server will be idle most of the time, and interdeparture times must be approximately distributed as that interarrival times. Based on this, we interpolate between these two extremes to get

$$C^2_{d,i} \approx (1-\rho_i)^2 C^2_{a,i} + \rho_i^2 C^2_{s,i}. \tag{2.3}$$

2.3.5. What is C^2_d for the $D/D/1$ queue according to (2.3)?

2.3.6. What is C^2_d for the $M/M/1$ queue according to (2.3)?

2.3.7. Consider two $G/G/1$ station in tandem. Suppose $\lambda = 2$ per hour, $C^2_{a,1} = 2$ at station 1, $C^2_s = 0.5$ 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 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.

Hints

h.2.3.1 Focus on the waiting times for each station separately, and realize that each is an $M/M/1$ queue. What is the arrival process at the second station? Recall Burke's law, c.f., 1.10.11.

h.2.3.2 If we can reduce all service variability at the second server, the second station can be modeled as an $M/D/1$ queue.

h.2.3.3 Realize that now also the distribution of interdeparture times of the first station changes and becomes more regular.

h.2.3.4 What would you do if there would be third station in this tandem network?

Solutions

s.2.3.1 The first queue is the familiar $M/M/1$ queue. (Why?) For the sequel it is important to observe that the departure process of the first station, i.e., the distribution of times between jobs leaving the first station, is the same as the arrival process. Consequently, the inter-departure times are also exponentially distributed with parameter λ. (However, the service times are exponentially distributed with parameter μ_1.)

What can we say about the second station? Clearly, the jobs departing at the first stations are the arrivals at the second station. Hence, the departure process being exponential with rate λ, the inter-arrival times at the second station are also exponential with rate λ. Consequently, the second queue is also an $M/M/1$ queue.

The total waiting time in the system, i.e., the time spent in both queues, is the sum of the waiting times at the first and second station. As each is an $M/M/1$ queue, the total waiting time has the form:

$$\begin{aligned} \mathsf{E}\left[W_Q\right] &= \mathsf{E}\left[W_{Q,1}\right] + \mathsf{E}\left[W_{Q,2}\right] \\ &= \frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}, \end{aligned} \tag{2.4}$$

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

s.2.3.2 As we reduce the variability of the second server, the service process is no longer exponential. However, the arrival process at the second station is still Poisson. As a consequence, the queueing discipline changes to the $M/G/1$ queue. The expected waiting time for this case has the form:

$$\mathsf{E}\left[W_{Q,2}\right] = \frac{1+C_{s,2}^2}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}, \tag{2.5}$$

where $C_{s,2}^2$ is the squared coefficient of variation of the service process of the second server.

By assumption we can entirely remove the variability of the second server. This yields that the coefficient of variation $C_{s,2}^s = 0$. Thus, the service process being deterministic, the second station becomes the $M/D/1$ queue.

The expected waiting time for the $M/D/1$ queue follows immediately from (2.5) by setting $C_{s,2}^2 = 0$:

$$\mathsf{E}\left[W_{Q,2}\right] = \frac{1}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}. \tag{2.6}$$

Clearly, this is half the waiting time of the $M/M/1$ queue.

Since we do not change the first station in any way, this is still an $M/M/1$ queue. Thus, the total time in queue for this scenario becomes:

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

s.2.3.3 Analogous to the previous situation, suppose we can set the coefficient of variation $C_{1,s}^2$ of the first server to zero. Thus, this becomes an $M/D/1$ queue, so that, similar to (2.6):

$$\mathsf{E}\left[W_{Q,1}\right] = \frac{1}{2}\frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1}.$$

Contrary to the $M/M/1$ queue, the inter-departures of the $M/D/1$ queue are not exponentially distributed. When the first server is busy, they are deterministic. When the first server is idle, we first need to wait for the next arrival, which is exponentially distributed, and then for this arrival to finish its service, which takes D. Thus, the time to the next departure is $X + D$.

However, for the sake of simplicity, let us simply assume in the sequel of this example that the departure process *is* deterministic.

As we previously remarked, the departure process of the first station forms the arrival process at the second station. Since the departures are assumed to be deterministic, the arrivals at the second station are also deterministic. The service times at the second station, however, are still exponential. Thus, the second station can be modeled as the $D/M/1$ queue. For this queue we need to derive an expression for the waiting time. The simplest approximation follows from an expression for the waiting time of the $G/G/1$ queue.

We know that the expected waiting time for the the $G/G/1$ queue has the approximate form:

$$\mathsf{E}\left[W_{Q,2}\right] = \frac{C_{a,2}^2 + C_{s,2}^2}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}, \tag{2.7}$$

Clearly, for our case, the coefficient of variation $C^2_{a,s}$ of the arrival process becomes, approximately, 0, while $C^2_{s,2} = 1$, since the service process is still exponential. Hence,

$$\mathsf{E}\left[W_{Q,2}\right] = \frac{1}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}. \tag{2.8}$$

Combining (2.7) and (2.8), the total time in queue becomes:

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

which is half the waiting time of the two $M/M/1$ stations in tandem, but also smaller than the situation in which we reduce the variability at the second station.

s.2.3.4 As a general guide line, it seems best to reduce the variability at the first station. The main point to remember is that reducing the variability of the service process at the first station also reduces the variability of its departure process, hence the variability of the arrival processes at the second station. Thus, the situation improves at two locations of the chain of stations, rather than at one.

s.2.3.5 Since the interarrival times and the service times are deterministic, $C^2_a = C^2_s = 0$. Hence $C^2_d = 0$.

s.2.3.6 Since the interarrival times and the service times are exponential, $C^2_a = C^2_s = 1$. Hence $C^2_d = 1$. This is precisely in line with our earlier insights, in which we obtained that the departure process of an $M/M/1$ queue is Poisson.

s.2.3.7 First station 1.

```
>>> labda = 2.
>>> S1 = 20./60
>>> rho1 = labda*S1
>>> rho1
0.6666666666666666
>>> ca1 = 2.
>>> cs1 = 0.5
>>> EW1 = (ca1+cs1)/2 * rho1/(1-rho1) * S1
>>> EW1
0.8333333333333331
>>> W1 = EW1 + S1
>>> W1
1.1666666666666665
```

Now station 2. We first need to compute C_{d1}^2.

```
>>> cd1 = (1-rho1**2)*ca1 + rho1**2*cs1
>>> cd1
1.3333333333333335

>>> labda = 2
>>> S2 = 25./60
>>> rho2 = labda*S2
>>> rho2
0.8333333333333334
>>> ca2 = cd1 # here we use our formula
>>> cs2 = 0.5
>>> EW2 = (ca2+cs2)/2 * rho2/(1-rho2) * S2
>>> EW2
1.909722222222223
>>> W2 = EW2 + S2
>>> W2
2.3263888888888897
```

With Little's law.

```
>>> W=W1+W2
>>> W
3.4930555555555562
>>> L = labda*W
>>> L
6.9861111111111125
```

3 Closed Queueing Networks

3.1 Gordon-Newell Networks

Theory and Exercises

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.

It is perhaps the easiest to start with studying the MVA algorithm, then the MDA algorithm and finish with the convolution algorithm.

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 $VP = V$.

3.1.1. 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.

3.1.2. Zijm.Ex.3.1.1

3.1.3. Zijm.Ex.3.1.2

3.1.4. Zijm.Ex.3.1.3

3.1.5. Relate Zijm.Eq.3.3 to the form of the steady-state distribution of the number of jobs in an $M/M/c$ queue.

3.1.6. Zijm.Ex.3.1.4

3.1.7. Zijm.Ex.3.1.5

Hints

h.3.1.7 First write down all different states, and then use Zijm.Eq.3.2 and 3.3.

Solutions

s.3.1.1 The routing matrix P is

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

Solving $V = VP$ leads to $(V_0, V_1, V_2) = (V_2/2, V_0 + V_2/2, V_1)$. Thus, from the last item, $V_2 = V_1$, and from the first $V_0 = V_2/2$. Since $V_0 = 1$, it follows that $V_2 = 2V_0$ and $V_1 = V_2 = 2V_0$.

s.3.1.2 If a part would need refitting or repositioning at the load/unload station, the part would visit the load/unload station more than once during its stay in the network. The visit ratio of the load/unload station can then no longer be set to 1.

s.3.1.3 Lets number the states from 1 to 4. If station 1 feeds into station 2, and so on, and station 4 into station 1, then

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

The visit ratios are, of course, $V_1 = V_2 = V_3 = V_4$.

s.3.1.4 We number station a as 1, and so on.

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1/5 & 0 & 4/5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We now solve for V in the equation $V = VP$. Since, by definition, $V_1 = 1$, it suffices to express the other visit ratios in terms of V_1. From elementary linear algebra, the second column implies that $V_2 = V_1$, and the third that $V_3 = 4/5V_2$, so that $V_3 = 4/5V_1$. From the fourth column $V_4 = 1/3V_3$, thus, $V_4 = 1/3 \cdot 4/5V_1 = 4/15V_1$. From the fifth column $V_5 = 2/3V_4$, hence $V_5 = 8/45V_1$. Finally, from the sixth column, $V_6 = V_4 + V_5$, hence $V_6 = (4/15 + 8/45)V_1$.

For later courses on Markov chains, it is important to note the following. Write the visit ratio equation $V = VP$ as $V(1-P) = 0$ where 1 is the indicator matrix. Clearly, V is must be a left eigenvector of the matrix $1-P$ with eigenvalue 0 (recall that $V(1-P) = 0 = 0 \cdot V$). Thus, at least one row or column of the matrix $1-P$ is superflous to solve for V.

s.3.1.5 Except for the normalization constant, the expressions are the same as equations 1.36 and 1.37 of Zijm's book.

s.3.1.6 Consider network one with routing $1 \to 3 \to 4 \to 2 \to 1$, each transition occuring with probability one, i.e.,

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

All nodes are is visited equally often. Another network could correspond to the routing $1 \to 2 \to 3 \to 4 \to 1$. Yet another would be this:

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

Observe that I choose the rate such that each node receives the same fraction of traffic.

For the interested: There must be many such 'equivalent' networks, but a general method to classify all equivalent networks seems hard. The question comes down to finding all stochastic matrices P that have at least one (left) eigenvector V in common. Observe that the visit ratio equation $VP = P$ implies that V is a left eigenvector with eigenvalue 1.

s.3.1.7 Since $M = 2$, we have three stations: Stations 0, 1, and 2. We also have $N = 2$ jobs. Thus, the states are $(2,0,0)$ (meaning that the load/unload station has 2 jobs, and stations 1 and 2 no jobs), $(0,2,0)$, $(0,0,2)$, $(1,1,0)$, $(1,0,1)$, and $(0,1,1)$. Note that the routing matrix P is not given, so that we cannot compute the visit ratios. Hence I leave them unspecified. Now, realize that for a fixed $\vec{n} = (n_0, n_1, n_2)$, $\Pi_i f_i(n_i) = f_0(n_0) f_1(n_1) f_2(n_2)$, so that if

$$\vec{n} = (2,0,0) \qquad f_0(2) f_1(0) f_2(0) = \left(\frac{V_0}{\mu_0}\right)^2,$$

$$\vec{n} = (0,0,2) \qquad f_0(0) f_1(0) f_2(2) = \left(\frac{V_2}{\mu_2}\right)^2,$$

$$\vec{n} = (0,2,0) \qquad f_0(0) f_1(2) f_2(0) = \frac{1}{2}\left(\frac{V_1}{\mu_1}\right)^2.$$

Note that in this last result we use that Station 1 has two servers. For the other combinations,

$$\vec{n} = (1,1,0) \qquad f_0(1) f_1(1) f_2(0) = \frac{V_0}{\mu_0}\frac{V_1}{\mu_1},$$

$$\vec{n} = (1,0,1) \qquad f_0(1) f_1(0) f_2(1) = \frac{V_0}{\mu_0}\frac{V_2}{\mu_2},$$

$$\vec{n} = (0,1,1) \qquad f_0(0) f_1(1) f_2(1) = \frac{V_1}{\mu_1}\frac{V_2}{\mu_2}.$$

Finally, add up all the above numbers to make $G(M,N)$:

3.2 MVA Algorithm

Theory and Exercises

3.2.1. 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.

3.2.2. Zijm.Ex.3.1.13. Assume that all stations have just one server.

3.2.3. Zijm.Ex.3.1.14

3.2.4. Implement the MVA algorithm in your preferred computer language and make Figure 3.2.

Hints

h.3.2.1 Start with $n = 1$, then consider $n = 2$ and so on.

Solutions

s.3.2.1 From $VP = V$ we conclude that $V_1 = 2V_0$. Take $n = 1$.

$$\begin{aligned}
\mathsf{E}[W_0(1)] &= \mathsf{E}[L_0(0) + 1]\,\mathsf{E}[S_0] = \mathsf{E}[S_0] = 2, \\
\mathsf{E}[W_1(1)] &= \mathsf{E}[L_1(0) + 1]\,\mathsf{E}[S_1] = \mathsf{E}[S_1] = 3, \\
\mathsf{E}[W(1)] &= \sum_{i=0}^{1} V_i\,\mathsf{E}[W_i(1)] = V_0 2 + V_1 3 = 1 \cdot 2 + 2 \cdot 3 = 8 \\
\mathrm{TH}_0(1) &= \frac{1}{\mathsf{E}[W(1)]} = \frac{1}{8}, \\
\mathsf{E}[L_0(1)] &= \mathrm{TH}_0(1)\,\mathsf{E}[W_0(1)] = \frac{2}{8} = \frac{1}{4} \\
\mathsf{E}[L_1(1)] &= \mathrm{TH}_1(1)\,\mathsf{E}[W_1(1)] = V_1 \mathrm{TH}_0(1)\,\mathsf{E}[W_1(1)] = 2 \cdot \frac{1}{8} \cdot 3 = \frac{3}{4}.
\end{aligned}$$

Now take $n = 2$.

$$\mathsf{E}[W_0(2)] = (\mathsf{E}[L_0(1) + 1]\,\mathsf{E}[S_0] = (1/4 + 1)\,\mathsf{E}[S_0] = \frac{5}{4} \cdot 2 = \frac{5}{2},$$

$$\mathsf{E}[W_1(2)] = (\mathsf{E}[L_1(1)+1]\,\mathsf{E}[S_1] = (3/4+1)\,\mathsf{E}[S_1] = \frac{7}{4}\cdot 3 = \frac{21}{4},$$

$$\mathsf{E}[W(2)] = \sum_{i=0}^{1} V_i\,\mathsf{E}[W_i(2)] = \frac{5}{2} + 2\cdot\frac{21}{4} = 13$$

$$\mathrm{TH}_0(2) = \frac{2}{\mathsf{E}[W(2)]} = \frac{2}{13},$$

$$\mathrm{TH}_1(2) = V_1\mathrm{TH}_0(2) = \frac{4}{13},$$

$$\mathsf{E}[L_0(2)] = \mathrm{TH}_0(2)\,\mathsf{E}[W_0(2)] = \frac{2}{13}\cdot\frac{5}{2} = \frac{5}{13}$$

$$\mathsf{E}[L_1(2)] = \mathrm{TH}_1(2)\,\mathsf{E}[W_1(2)] = \frac{4}{13}\cdot\frac{21}{4} = \frac{21}{13}.$$

And so on.

s.3.2.2 Since there are quite a lot of jobs, and station 0 is the bottleneck, i.e., the station with the highest load, most of the time there will be a queue at station 0. In fact, most of the jobs will be in queue in front of station 0. Therefore the rate out of station 0 will be approximately equal to its service rate, i.e., μ_0. Thus, station 1 will receive jobs at a rate $\lambda_1 \approx \mu_0$, i.e., $\lambda_1 \approx 2$. Now I simply approximate the queueing process at station 1 as an $M/M/1$ queue with $\mu_1 = 2.5$. Hence, $\rho_1 = \lambda_1/\mu_1 = 2/2.5 = 4/5$, hence $L_1 = \rho_1/(1-\rho_1) = (4/5)/(1/5) = 4$.

s.3.2.3 I use the MVA algorithm to compute the expected number of jobs at each of the stations. Again, as in an earlier question in which I implemented the convolution algorithm, nearly as a one-liner, studying the code in detail is very rewarding. I include the numerical results at the end.

Start with computing the visit ratios.

```
>>> from functools import lru_cache
>>> import numpy as np
>>> from numpy.linalg import eig

>>> mu = np.array([2, 2.5, 3, 3, 3])
>>> c = np.array([1, 1, 1, 1, 1])  # single server stations
>>> P = np.matrix([
...     [0, 1, 0, 0, 0],
...     [0, 0, 1, 0, 0],
...     [0, 0, 0, 1, 0],
```

```
...     [0, 0, 0, 0, 1],
...     [1, 0, 0, 0, 0],
... ]
... )

>>> v, w = eig(P.T)
>>> x = v.real.argmax()
>>> V = w[:, x].real
>>> V = V / V[0]
>>> print(V)
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

The visit ratios are according to expectation.

The following implements the MVA algorithm. Note how easy the code becomes with memoization; I only have to specify the recursions and the 'boundary conditions', i.e., what to do when $n = 0$. For the rest I don't have to think about in what exact sequence each of the functions needs to be called. The memoization takes care of all these problems.

```
>>> @lru_cache(maxsize=None)
... def EL(j, n):
...     if n <= 0:
...         return 0
...     else:
...         return TH(j, n) * EW(j, n)
...

>>> @lru_cache(maxsize=None)
... def EW(j, n):
...     return (EL(j, n - 1) + 1) / mu[j]
...

>>> @lru_cache(maxsize=None)
... def TH(j, n):
...     if j == 0:
```

```
...         return n / sum(V[j] * EW(j, n) for j in range(len(mu)))
...     else:
...         return V[j] * TH(0, n)
...
```

This is all!

Now I print the expected jobs at the stations.

```
>>> print("  n    EL0    EL1    EL2   EL3    EL4    TH0")
  n    EL0    EL1    EL2   EL3    EL4    TH0
>>> for n in range(1,30):
...     res = "{:3d}".format(n)
...     for j in range(len(mu)):
...         res += "{:6.2f}".format(float(EL(j, n)))
...     res += "{:6.2f}".format(float(TH(0,n)))
...     print(res)
...
  1  0.26  0.21  0.18  0.18  0.18  0.53
  2  0.55  0.42  0.34  0.34  0.34  0.87
  3  0.87  0.64  0.50  0.50  0.50  1.12
  4  1.21  0.85  0.65  0.65  0.65  1.30
  5  1.58  1.06  0.79  0.79  0.79  1.43
  6  1.99  1.27  0.92  0.92  0.92  1.54
  7  2.42  1.47  1.04  1.04  1.04  1.62
  8  2.89  1.67  1.15  1.15  1.15  1.69
  9  3.39  1.86  1.25  1.25  1.25  1.74
 10  3.93  2.05  1.34  1.34  1.34  1.79
 11  4.50  2.23  1.42  1.42  1.42  1.83
 12  5.10  2.40  1.50  1.50  1.50  1.86
 13  5.74  2.56  1.57  1.57  1.57  1.88
 14  6.41  2.70  1.63  1.63  1.63  1.90
 15  7.11  2.84  1.68  1.68  1.68  1.92
 16  7.84  2.97  1.73  1.73  1.73  1.93
 17  8.60  3.09  1.77  1.77  1.77  1.95
 18  9.39  3.20  1.80  1.80  1.80  1.96
 19 10.20  3.30  1.84  1.84  1.84  1.96
 20 11.03  3.39  1.86  1.86  1.86  1.97
 21 11.88  3.47  1.88  1.88  1.88  1.98
 22 12.75  3.54  1.90  1.90  1.90  1.98
```

```
 23 13.64  3.60  1.92  1.92  1.92  1.98
 24 14.54  3.66  1.93  1.93  1.93  1.99
 25 15.46  3.70  1.95  1.95  1.95  1.99
 26 16.39  3.75  1.96  1.96  1.96  1.99
 27 17.33  3.78  1.96  1.96  1.96  1.99
 28 18.27  3.82  1.97  1.97  1.97  1.99
 29 19.23  3.84  1.98  1.98  1.98  2.00

now exiting Console...
```

This is interesting. Using the insights of the previous question, approximation stations 3, 4 and 5 as $M/M/1$ queues, we have according to this model that $\mathsf{E}[L_3] = \rho_3/(1-\rho_3)$. Taking $\lambda_3 = 2$ and $\mu_3 = 3$, we see that $\rho_3 = 2/3$ so that $\mathsf{E}[L_3] = 2$. This is very close to 1.98.

For the last line, with $n = 29$, the number of jobs at station 0 must therefore be approximately $29 - 4 - 2 - 2 - 2 = 19$. The algorithm supports our intuition!

s.3.2.4 See the answer of the previous question.

4 Notation

a_n = Number of arrivals in the nth period

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

A_k = Arrival time 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 interarrival 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_n = Departure time of nth job

$\tilde{A}_n$ = Departure time of the queue of nth 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 events in $[0,t]$

$N(s,t)$ = Number of events in $(s,t]$

$p(n)$ = Long-run time average that the system contains n jobs

Q_n = Queue length as seen by the nth job, or at the *end* of the nth 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_n = time in the system of nth job

$W_{Q,n}$ = time in the queue of nth job

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

$\mathsf{E}[W]_Q$ = sample average of the time in queue

X_k = inter-arrival time between job $k-1$ and job k

X = generic interarrival 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

5 Formula Sheet

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

$$C_{sB}^2 = \frac{\mathsf{V}[S_s] + N\,\mathsf{V}[S_0]}{(\mathsf{E}[S_s] + N\,\mathsf{E}[S_0])^2}$$

$$\sigma_e^2 = \sigma_0^2 + \frac{\sigma_s^2}{N_s} + \frac{N_s - 1}{N_s^2}(\mathsf{E}[S_s])^2$$

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

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

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

$$Q_{ij} = \frac{\lambda_{ij}}{\lambda_j}, i = 0 \ldots M, j = 1 \ldots M$$

$$w_j = (1 + 4(1-\rho_j)^2(v_j - 1))^{-1}$$

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

$$G(m,n) = \sum_{k=0}^{n} f_m(k)G(m-1, n-k)$$

$$\mathsf{E}\left[W_j(n)\right] = \mathsf{E}\left[L_j(n-1) + 1\right]\mathsf{E}\left[S_j\right]$$

$$\mathrm{TH}_0(n) = \frac{n}{\sum_{j=0}^{M} V_j\,\mathsf{E}\left[W_j(n)\right]}$$

$$p_j(0|0) = 1$$

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

$$C_e^2 = C_0^2 + 2A(1-A)\frac{}{\mathsf{E}}$$

$$\sigma_e^2 = \sigma_0^2 + \sigma_r^2 f_r + f_r(1-f_r)(\mathsf{E}[S$$

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

$$C_{si}^2 = \max\{C_{si}^2$$

$$C_{ij}^2 = P_{ij}C_{di}^2 + 1 -$$

$$C_{aj}^2 = w_j\sum_{i=0}^{M} Q_{ij}C_{ij}^2 + 1$$

$$v_j = \left(\sum_{i=0}^{M} Q_{ij}^2\right)^{-1}, j = 1, \ldots$$

$$V_i = (VP)_i = \sum_{j=0}^{M} V$$

$$G(m,0) = 1, G(0,n) =$$

$$\mathsf{E}\left[L_j(n)\right] = \mathrm{TH}_j(n)\,\mathsf{E}\left[W_j\right.$$

$$\mathrm{TH}_j(n) = V_j\mathrm{TH}$$

$$\mathsf{E}\left[W_j(n)\right] = \sum_{k=c_j}^{n-1}\frac{k - c_j + 1}{c_j\mu_j}p_j(k|n-1)$$

$$p_j(k|n) = \frac{\mathrm{TH}_j(n)}{\mu_j \min\{c_j, k\}} p_j(k-1|n-1), k = 1 \ldots N \qquad p_j(0|n) = 1 - \sum_{k=1}^{n} p_j($$

Bibliography

F. Baccelli and W.A. Massey. A sample path analysis of the $M/M/1$ queue. *Journal of Applied Probability*, 26(2):418–422, 1988.

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.

W.J. Hopp and M.L. Spearman. *Factory Physics*. Waveland Press, Inc., 3rd edition, 2008.

H. Sakasegawa. An approximation formula $l_q = \alpha\beta^\rho/(1-\rho)$. *Ananals of the Institute for Statistical Mathematics*, 29:67–75, 1977.

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.