Analysis of Queueing Systems with Sample Paths and Simulation

Nicky D. van Foreest

February 11, 2021

CONTENTS

1

INTRODUCTION

In the first section of this chapter we provide some high level motivation why to study queueing systems and an overview of the book. In the second we recall some notation and results of analysis and probability that we use throughout the book.

1.1 MOTIVATION AND OVERVIEW

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 gets his/her groceries for free when there is a queue of 4 or more customers and there is an unoccupied cashier desk. The manager that controls the occupation of the cashier positions is keen on keeping the queue small. Now 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 is to control the service capacity such the penalties and the personnel cost are about equally large.

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

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

CLEARLY, IN ALL these examples, the performance of the queueing system has to be monitored and controlled. Typically the following performance measures are relevant.

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.

In this book, we will be primarily concerned with making models of queueing systems such that we can compute or estimate these performance measures.

IN CHAPTER 2 WE start with constructing queueing systems in discrete time and continuous time. This serves three goals. First, construction is intellectually quite concrete, so that by specifying the rules to characterize the behavior of the system, you (the reader) develop essential modeling skills. Second, these rules can often be easily implemented in computer code, hence used to simulate actual queueing system. Simulation is in general the best way to analyze practical queueing systems, as realistic systems seldom yield to mathematics. Third, we will use sample-path arguments to develop the theoretical results of Chapter 6 and onwards, and these sample paths are precisely what simulators produce.

NOTWITHSTANDING THE POWER of simulation, it is often hard to obtain structural understanding of the behavior of queueing systems. Instead, mathematical models, whether exact or approximate, are useful to help reason about and improve queueing systems. In Chapter 4 we use approximations and general results of probability theory to understand how production and service situations are affected by the system parameters such as service speed, batching rules, and outages. As such, the first two chapters illustrate and motivate the study of practical queueing systems,

ONCE IT IS clear what queueing theory is about, the stage is set for a more mathematical treatment of queueing systems. In Chapter 5 we develop some key results that we need for this. We use sample paths of queueing process as a guiding principle, and assume that sample paths capture the 'normal' stochastic behavior of the system (with probability one.) Since we can also construct sample paths with simulation, sample paths form a direct bridge between the practical aspects of queueing theory of Chapter 2 and the mathematical analysis of the rest of the book.

IN CHAPTER 6 WE develop exact models for single-station queueing systems and in Chapter 7 we extend our study to simple examples of queueing control and queueing networks. Here we combine the material of the earlier chapters with other mathematical tools such as difference and differential equations, and non-negative matrices. As such, the last chapter in particular provides a stepping stone to Markov chains, Markov decision theory, optimization, dynamic programming, and so on.

IN OUR DISCUSSIONS we mostly focus on obtaining an intuitive understanding of the analytical tools. For proofs and/or more extensive results we refer to the bibliography at the end of the book.

THE MAIN TEXT contains a some examples or derivations, but most of this work have been delegated to exercises. Some exercises consist of consistency checks between results derived for different queueing models, and thereby provide important relations between various parts of the text. We note in passing that, while such checks are trivial in principle, the algebra can be quite difficult at times. The exercises are not meant to be really easy; they should require (some) work. Hints and solutions to all problems. are available at the end of the book.

1.2 INDISPENSABLE KNOWLEDGE

THERE ARE SOME fundamental concepts you are supposed to have seen earlier in your career. We recall them here, and use them over and over in the rest of the book, mostly without reference.

WE INTRODUCE THE following notation:

$$\mathbb{1}_A = \begin{cases} 1, & \text{if } A \text{ is true,} \\ 0, & \text{if } A \text{ is false.} \end{cases}, \qquad [x]^+ = x\,\mathbb{1}_{x\geq 0}, \tag{1.2.1}$$

$$f(x-) = \lim_{y\uparrow x} f(y), \qquad f(x+) = \lim_{y\downarrow x} f(y). \tag{1.2.2}$$

The (set) function $\mathbb{1}$ is known as the indicator function[0].

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

You should know that:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^{n-i} b^i, \tag{1.2.3a}$$

$$e^x = \lim_{n\to\infty} (1 + x/n)^n, \tag{1.2.3b}$$

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}, \tag{1.2.3c}$$

$$\sum_{n=0}^{N} \alpha^n = \frac{1-\alpha^{N+1}}{1-\alpha}. \tag{1.2.3d}$$

FOR A NON-NEGATIVE, integer-valued random variable X with probability mass function[2] $f(k) = \mathsf{P}\{X = k\}$, distribution function[3] $F(k) = \mathsf{P}\{X \leq k\}$ and survivor function[4] $G(k) = \mathsf{P}\{X > k\}$ we have

[0] *Indicator function*

[1] *Small o notation*

[2] *Probability mass function*

[3] *Distribution function*

[4] *Survivor function*

$$X = \sum_{n=0}^{\infty} X \mathbb{1}_{X=n} = \sum_{n=0}^{\infty} n \mathbb{1}_{X=n}, \tag{1.2.4a}$$
$$\mathsf{E}[\mathbb{1}_{X\leq x}] = \mathsf{P}\{X \leq x\}, \tag{1.2.4b}$$
$$\mathsf{E}[X] = \mathsf{E}\left[\sum_{n=0}^{\infty} n \mathbb{1}_{X=n}\right] = \sum_{n=0}^{\infty} n \mathsf{E}[\mathbb{1}_{X=n}] = \sum_{n=0}^{\infty} n f(n), \tag{1.2.4c}$$
$$\mathsf{E}[g(X)] = \sum_{n=0}^{\infty} g(n) f(n), \tag{1.2.4d}$$
$$\mathsf{V}[X] = \mathsf{E}\left[X^2\right] - (\mathsf{E}[X])^2. \tag{1.2.4e}$$
$$\tag{1.2.4f}$$

Eq. (1.2.4d) is known as the law of the unconscious statistician[5].

[5] *Law of the unconscious statistician*

Let X be a continuous non-negative random variable with distribution function F. We write $\mathsf{E}[X] = \int_0^\infty x F(\mathrm{d}x)$ for the expectation of X. Here $F(\mathrm{d}x)$ acts as a (sort of) shorthand for $f(x)\mathrm{d}x$.[6] Recall that $\mathsf{E}[g(X)] = \int_0^\infty g(x) F(\mathrm{d}x)$.

[6] *For details, see Capiński and Zastawniak [2003].*

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

For general random variables X and Y:

$$\mathsf{E}[X+Y] = \mathsf{E}[X] + \mathsf{E}[Y], \tag{1.2.5}$$

and, if X and Y are independent so that $\mathsf{E}[XY] = \mathsf{E}[X]\,\mathsf{E}[Y]$,

$$\mathsf{V}[X+Y] = \mathsf{V}[X] + \mathsf{V}[Y]. \tag{1.2.6}$$

THE MOMENT-GENERATING FUNCTION[7] $M_X(s)$ of a random variable X is defined for $s \in \mathbb{R}$ sufficiently small as:

[7] *Moment-generating function*

$$M_X(s) = \mathsf{E}\left[e^{sX}\right]. \tag{1.2.7a}$$

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

$$\mathsf{E}[X] = M_X'(0) = \left.\frac{\mathrm{d}M_X(s)}{ds}\right|_{s=0}, \tag{1.2.7b}$$
$$\mathsf{E}\left[X^2\right] = M_X''(0), \tag{1.2.7c}$$

and, if X and Y are independent,

$$M_{X+Y}(s) = M_X(s) \cdot M_Y(s). \tag{1.2.7d}$$

ABOUT CONDITIONAL PROBABILITY[8] you should know that

[8] *Conditional probability*

$$\mathsf{P}\{A \,|\, B\} = \frac{\mathsf{P}\{AB\}}{\mathsf{P}\{B\}}, \quad \text{if } \mathsf{P}\{B\} > 0, \tag{1.2.8a}$$

and, if $A = \bigcup_{i=1}^n B_i$ with $\mathsf{P}\{B_i > 0\}$ for all i,

$$\mathsf{P}\{A\} = \sum_{i=1}^{n} \mathsf{P}\{AB_i\} = \sum_{i=1}^{n} \mathsf{P}\{A \,|\, B_i\}\, \mathsf{P}\{B_i\}. \tag{1.2.8b}$$

Ex 1.2.1. Show that $|h|^\alpha = o(h)$ for all $\alpha > 1$. Conclude in particular that $ah^2 = o(h)$ for any constant a.

Ex 1.2.2. Let c be a constant (in $\mathbb{R}$) and the functions f and g both of $o(h)$. Then show that (1) $f(h) \to 0$ when $h \to 0$, (2) $c \cdot f = o(h)$, (3) $f + g = o(h)$, and (4) $f \cdot g = o(h)$.

Ex 1.2.3. Why is $e^x = 1 + x + o(x)$?

Ex 1.2.4. Why is (1.2.4a) true?

Ex 1.2.5. Show that $G(k) = \sum_{m=0}^{\infty} \mathbb{1}_{m>k} f(m)$ for discrete X.

Ex 1.2.6. For a nonnegative discrete random variable X, use indicator functions to prove that $\mathsf{E}[X] = \sum_{k=0}^{\infty} G(k)$.

Ex 1.2.7. For discrete X, use indicator functions to prove that $\sum_{i=0}^{\infty} iG(i) = \mathsf{E}\left[X^2\right]/2 - \mathsf{E}[X]/2$.

Ex 1.2.8. For general non-negative X, use indicator functions to prove that $\mathsf{E}[X] = \int_0^\infty xF(\mathrm{d}x) = \int_0^\infty G(y)\,\mathrm{d}y$, where $G(x) = 1 - F(x)$.

Ex 1.2.9. Use indicator functions to prove that for a general non-negative random variable X, $\mathsf{E}\left[X^2\right] = 2\int_0^\infty yG(y)\,\mathrm{d}y$.

Ex 1.2.10. Show that $\mathsf{E}\left[X^2\right]/2 = \int_0^\infty yG(y)\,\mathrm{d}y$ for a continuous non-negative random variable X with survivor function G.

Ex 1.2.11. What is the value of $M_X(0)$?

Ex 1.2.12. We have one gift to give to one out of three children. As we cannot divide the gift into parts, we decide to let 'fate decide'. That is, we choose a random number in the set $\{1, 2, 3\}$. The first child that guesses this number wins the gift. Show that the probability of winning the gift is the same for each child.

2

CONSTRUCTION AND SIMULATION OF QUEUEING SYSTEMS

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

In Section 2.1 we build discrete-time models of queueing systems, which means that we use the number of jobs that arrive and can be served in fixed periods of time to construct the queueing process. Such a period can be an hour, or a day; in fact, any amount of time that makes sense in the context in which the model will be used. Typically, we model the number of arrivals and potential services as random variables, and in many practical settings it is reasonable to take the number of arrivals in a period as Poisson distributed. This being the case, we consider the Poisson distribution in Section 2.2, and once we have an understanding of this process, we can use random number generators to generate (Poisson distributed) random numbers of arrivals and services to drive the simulator.

In Section 2.3 we focus on constructing queueing processes in continuous time. In this setting, the inter-arrival times and service times of individual jobs become of importance, and then exponentially distributed random variables play a fundamental role. We therefore discuss the properties of the exponential distribution in Section 2.4. There we also mention the interesting and close relationship between the exponential distribution and the Poisson distribution.

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

2.1 QUEUEING PROCESSES IN DISCRETE-TIME

We start with a case to provide motivation to study queueing systems. Then we develop a set of recursions of fundamental importance by which we can simulate the case and evaluate the efficacy of several policies to improve the system.

To illustrate the power of this approach, this section contains many exercises in which you are asked to model different queueing situations by means of such recursions. Once the recursions are obtained, the These recursions are often quite easy so that they can also be useful to discuss how well the model captures 'reality' with managers, medical doctors, and so on.

AT A MENTAL HEALTH department, five psychiatrists do intakes of future patients to determine the best treatment process once patients 'enter the system'. There are complaints about the time patients have to wait for the intake; the desired waiting time is around two weeks, but the realized waiting time is sometimes more than three months. This is unacceptably long, but ... what to do about it?

THE FIVE PSYCHIATRISTS put forward various suggestions to reduce the waiting times.

1. Give all psychiatrists the same weekly capacity for doing intakes. Motivation: Currently one psychiatrist does 9 intakes per week while two other psychiatrists do only 1. This is not a problem during weeks when all psychiatrists are present; however, psychiatrists take holidays, visit conferences, and so on. So, if the psychiatrist with the most intakes per week goes on leave, this affects the intake capacity considerably.

2. Synchronize holidays.[0] Motivation: to reduce the variation in the service capacity, the psychiatrists plan their holidays consecutively. However, perhaps it is better to work at full capacity or not at all.

3. Control[1] the intake capacity as a function of the waiting time. Motivation: in analogy with supermarkets, scale up (down) capacity when the queue is long (short).

WE NEXT DEVELOP a method to simulate the behavior of the system over time so that we can evaluate the effect of the above suggestions. We use simulation, because the mathematical analysis is too hard.[2]

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[3] and develop recursions for the behavior of the queue from period to period. Using fixed-sized periods has the advantage that we do not have to specify specific inter-arrival times or service times of individual customers; only the number of arrivals in a period and the number of potential services are relevant.

Let us define:

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

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

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

L_k = the system length[4], i.e., the number of jobs in the system at the *end* of period k.[5]

In the sequel we also call a_k the *size of the batch* arriving in period k. Note that the definition of a_k is a bit subtle: we may assume that the

[0] With the insights of Chapter 4 we can immediately see that this is a bad suggestion.

[1] People often object to such policies because they believe that they have to do more work. However, this is wrong. Suppose that 1000 patients per year need treatment. This number does not depend on whether they spend time in queue or not.

[2] In case of doubt, try to analyze transient multi-server queueing systems with vacations, of which this is an example.

[3] The length of these periods depends on context. For the present case, it is appropriate to take weeks. For supermarkets perhaps a length of 5 minutes is best.

[4] System length

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

arriving jobs arrive either at the start or at the end of the period. In the first case, the jobs can be served in period k, in the latter case, they *cannot* be served in period k.

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

$$d_k = \min\{L_{k-1}, c_k\}, \tag{2.1.1a}$$

Now that we know the number of departures, the number at the end of period k is given by

$$L_k = L_{k-1} - d_k + a_k. \tag{2.1.1b}$$

Like this, if we are given L_0, we can obtain L_1, and from this L_2, and so on.[6]

IT IS IMPORTANT to realize that the above recursions only construct $\{L_k\}$, i.e., the dynamics of the number of jobs in the system. If we also need information about the sojourn times[7], i.e., the time jobs spend in the system, it is necessary to specify the service discipline[8], i.e., a rule that decides on the order in which jobs in queue are taken into service.[9] In this book we assume henceforth that jobs move to the server in the order in which they arrive. This is known as First-In-First-Out (FIFO)[10]; First-Come-First Serve (FCFS) is another much used acronym. There are many other rules, such as Last-In-First-Out, but we do not discuss these here.

WE NOTE THAT there is a difference between *waiting time* W and sojourn time J. The former is the time jobs spend in queue, the latter the time in the system, which is the waiting plus the time at the server L_s. In the model (2.1.1) we implicitly include service time, so that we should actually speak about sojourn time, which we henceforth do in this section.

IT IS CLEAR that in (2.1.1) we assume that jobs that arrive in period k cannot be served in period k. If the situation is such that jobs *can* be served in the same period as they arrive, then (2.1.1) should be changed to[11]

$$d_k = \min\{L_{k-1} + a_k, c_k\}. \tag{2.1.2}$$

Which of (2.1.1) or (2.1.2) to choose depends on the context, and what we like to model. If we like to be 'on the safe side', then it is perhaps best to use (2.1.1) because with this rule, we overestimate the queue lengths, while with (2.1.2) we underestimate the queue lengths. In general, there is no rule is 'best'.

OF COURSE WE are not going to carry out the computations for $\{L_k\}$ by hand. Typically, we use company data to model the arrival process $\{a_k\}$ and the capacity $\{c_k\}$, and feed this data into a computer to carry out the recursions (2.1.1). If we do not have sufficient data, we make a probability model for these data and use the computer to generate random numbers with, hopefully, similar characteristics as the real data.

[6] *In a sense, (2.1.1) is the $F = ma$ of a queueing system: Given initial conditions, we apply the rule at time $k-1$ to get the state at time k, and so on.*

[7] *Sojourn times*

[8] *Service discipline*

[9] *[2.1.9]*

[10] *First-In-First-Out (FIFO)*

[11] *[2.1.2]*

At any rate, from this point on, we assume that it is easy, by means of computers, to obtain numbers $a_1,\dots,a_n$ for $n \gg 1000$, and so on.

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

AS A FIRST STEP in the analysis, we model the arrival process of patients as a Poisson process, see Section 2.2. The duration of a period is taken to be a week. The average number of arrivals per period, based on data of the company, was slightly less than 12 per week; in the simulation we set it to $\lambda = 11.8$ per week.

NEXT, WE MODEL the capacity in the form of a matrix such that row i corresponds to the weekly capacity of psychiatrist i:

$$C = \begin{pmatrix} 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ 1 & 1 & 1 & \dots \\ 3 & 3 & 3 & \dots \\ 9 & 9 & 9 & \dots \end{pmatrix}. \tag{2.1.3}$$

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:

$$C = \begin{pmatrix} 2 & 2 & 2 & \dots \\ 2 & 2 & 2 & \dots \\ 3 & 3 & 3 & \dots \\ 4 & 4 & 4 & \dots \\ 4 & 4 & 4 & \dots \end{pmatrix}. \tag{2.1.4}$$

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

$$C = \begin{pmatrix} 0 & 2 & 2 & 2 & 2 & 0 & \dots \\ 2 & 0 & 2 & 2 & 2 & 2 & \dots \\ 3 & 3 & 0 & 3 & 3 & 3 & \dots \\ 4 & 4 & 4 & 0 & 4 & 4 & \dots \\ 4 & 4 & 4 & 4 & 0 & 4 & \dots \end{pmatrix}. \tag{2.1.5}$$

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 (2.1.1) to simulate the queue length process for the four different scenarios proposed by the psychiatrists, unbalanced versus balanced capacity, and spread out holidays versus simultaneous holidays.

The results are shown in Fig. 1. We plot, for each period, the largest and the smallest queue that occurred under all four capacity plans that result from following the first and second suggestions of the psychiatrists. The graphs make clear that these suggestions hardly affect the behavior of the queue length process.

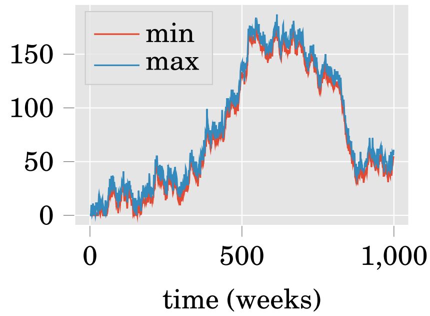


Figure 1: Effect of capacity and holiday plans. Per time point we plot the maximum and the minimum queue length under the policies.

Now we consider Suggestion 3, which comes down to doing more intakes when it is busy, and fewer when it is quiet. A simple rule is to let the capacity for week k depend on the queue length of the week before, for instance,

$$c_k = \begin{cases} 12+e, & \text{if } L_{k-1} \geq 24, \\ 12-e, & \text{if } L_{k-1} \leq 12. \end{cases} \tag{2.1.6}$$

We can take $e = 1$ or 2, or perhaps a larger number; the larger e, the larger the control we exercise. We can of course also adapt the thresholds 12 and 24.

Let's consider three different control levels, $e = 1$, $e = 2$, and $e = 5$; when $e = 5$, each psychiatrist does one extra intake. The results, see Fig. 2, show a striking difference indeed. The queue does not explode anymore; already taking $e = 1$ has a large influence.

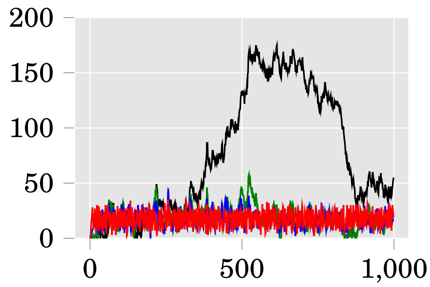


Figure 2: Controlling the number of intakes.

THIS SIMULATION EXPERIMENT shows that changing holiday plans or spreading the work over multiple servers, i.e., psychiatrists, may not significantly affect the queueing behavior. However, controlling the service rate as a function of the queue length can improve the situation dramatically.

IN GENERAL, with recursions as (2.1.1) we can carry out simple what-if-analyses. For instance, a hospital is considering to buy a second MRI scanner. The current one is saturated, as it is used from 8 am to 6 pm, and can certainly not serve the forecasted demand. But, suppose we can find a percentage of staff willing to work from 8 pm to 11 pm, say, and that patients, 30% perhaps, are also prepared to come to the hospital for an MRI scan.[12] This idea would increase the capacity with about 30% = (3/(18 − 8)) thereby enabling the hospital to postpone the investment in an MRI scanner with one or two years.

[12] *Perhaps some staff and patients even prefer to work/have scans in the evening.*

THE READER SHOULD understand from the above case that, once we have recursions such as (2.1.1) and control rules such as (2.1.6), we can simulate the system and make plots to evaluate suggestions for improvement. Therefore most of the exercises below ask to come up with recursions for specific queueing systems.

As an aside, it may be that the recursions you find are not identical to the recursions of the solutions, because the assumptions you make are slightly different from the ones I make. In fact, only the recursions completely specify the model, but if the problem statement would contain the recursions, there would be nothing left to practice anymore.

Ex 2.1.1. Suppose that $c_k = 7$ for all k, $a_1 = 5$, $a_2 = 4$ $a_3 = 9$, and $L_0 = 8$, compute L_3.

Ex 2.1.2. Prove that the scheme $L_k = [L_{k-1} + a_k - c_k]^+$ generates a system in which jobs can be served in the period they arrive.

Serve jobs in the period in which they arrive.

Ex 2.1.3. A queueing system is under periodic review, i.e., at the end of each period the queue length is measured. Jobs arriving at period k cannot be served in period k and the system cannot contain more than K jobs. Develop code to simulate $\{L_k\}$ and compute the amount of jobs lost per period, and the fraction of jobs lost after simulating for T periods.

Queue with blocking

Ex 2.1.4. A fraction p of the items produced in a period by a machine turns out to be faulty, and has to be made anew the next period. Develop a set of recursions to simulate $\{L_k\}$.

Systems with yield loss

Ex 2.1.5. A fraction p of items do not meet the quality requirements after the first pass at a machine, but require a second pass to repair the problems. Assume that repair jobs need half the service time of new jobs and are served with priority over new jobs. Develop the recursions.

Rework

Ex 2.1.6. Demand arriving at a single-server queue in a period can be served in that period. At the start of period k, the server capacity is set to c_k at cost βc_k. A job charges h per period in the system. Make a model that computes the cost for the first T periods.

Cost models

Ex 2.1.7. To ensure that the server capacity is always fully used, a server works at rate c in period k if $L_{k-1} \geq N$ jobs, and not otherwise. Set up the recursions.

Server controlled by threshold policy

Ex 2.1.8. A machine can switch on and off. If the system length hits N (becomes empty) in period k, the machine switches on (off) in period $k+1$. It costs K to switch on the machine. There is also a cost β per period when the machine is on, and it costs h per period per customer in the system. Make a model to computes the cost for the first T periods. Assume the machine is off at $k = 0$.

Cost of an N policy.

Ex 2.1.9. Take $d_k = \min\{L_{k-1} + a_k, c_k\}$, and assume that jobs are served in FIFO sequence. Find expressions for the shortest (longest) possible sojourn time $J_{-,k}$ ($J_{+,k}$) of a job that arrives at time k.

Estimating sojourn time, rather than queue length.

Ex 2.1.10. A machine serves with two queues such that jobs in the first queue get priority over jobs in the other queue. Find the recursions.

Priority queueing

Ex 2.1.11. One server serves two queues. Each queue receives service capacity in proportion to the queue length. Derive the recursions.

Proportionally fair queueing

Ex 2.1.12. Consider a single-server that serves two parallel queues. Queue i has minimal guaranteed service capacity r^i each period, such that $c_k \geq r^1 + r^2$. Extra capacity beyond the reserved capacity is given to queue 1 with priority. (An example is a psychiatrist who reserves capacity for different patient groups.) Formulate the recursions.

Queues with reserved service capacity

Ex 2.1.13. Consider a single-server that serves two parallel queues. Queue i receives a minimal service capacity r^i every period. Reserved capacity unused for one queue cannot be used to serve the other queue. Any extra capacity beyond the reserved capacity, i.e, $c_k - r^2$, is given to queue 1 with priority. Formulate the recursions.

Queue with protected service capacity and lost capacity

An example is the operation room of a hospital. There is a weekly capacity, part of which 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' often applies to service capacity.

Ex 2.1.14. Consider a production network with two production stations in tandem, that is, the jobs processed at station 1 move at the end of period k to station 2. What are the recursions?

Tandem networks

Ex 2.1.15. Consider a production network with two production stations in tandem and blocking: the server at station 1 is not allowed to produce more than station 2 can contain, i.e., $c^1_{k+1} \leq M - L^2_k$. Find recursions.

A tandem queue with blocking

Ex 2.1.16. Consider a production situation with two stations A and B that send their products to station C. Derive the recursions.

Merging streams

Ex 2.1.17. Consider a paint mixing machine that produces products for two downstream packaging machines A and B, each with its own queue. In the simplest model, the content of the queue at the mixing machine is proportional to the demands λ^A and λ^B for the packaging machines. Provide the recursions.

Splitting streams

Ex 2.1.18. 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, which requires one period setup time. Make a model.

Queues with setup times

An example: a nurse takes blood samples at two departments in a hospital. It takes time to walk from one location to another. An interesting, but hard question: what rule would minimize average waiting time?

2.2 POISSON DISTRIBUTION

In this section we first introduce the Poisson process to model the arrival process of jobs[0] that need to receive service at a station.[1]

In the exercises we derive numerous properties of the Poisson process; the rest of the book we will use these results time and again.

CONSIDER A STREAM OF CUSTOMERS that enter a shop during a time interval of duration $t = 1$ hour.[2] If we chop up the hour into small periods of 1 seconds, it is reasonable to assume that the probability of an arrival in a specific period is small and the probability of two or more arrivals is very small. Moreover, it is reasonable to assume that the probability that a customer enters in a certain second is the same for each second.

[0] A job can be anything that requires service; it can be a customer, an item to make, a car to repair, and so on.

[1] A station is a general name for a shop, a machine, a mechanic, and so on.

[2] The unit of time t depends on the context. For example, when we model a shop, it is reasonable to take t in the order of one hour, but when we think about a certain type of operations at a hospital, the t might be weeks.

More formally, to model such an arrival process, we make a few assumptions. We think about the interval as composed of n many small subintervals, all with equal length t/n. We take n so large that we can neglect the probability that two or more customers arrive in one small interval. Moreover, we assume that the arrival of a customer in a certain subinterval is independent of the (non)-arrival of customers in any of the other subintervals, and that the probability of an arrival is given by a (very) small probability p.

In precise terms, we model the arrivals of customers as a set of identically and independently distributed (i.i.d.)[3] random variables $\{B_i\}$ such that $B_i = 1$ denotes an arrival in interval i, and $B_i = 0$ no arrival. The arrival probability is $\mathsf{P}\{B_i = 1\} = p$.

Bernoulli distributed[4]

It is well-known that the total number of arrivals $N_n(t)$ that occur in the n intervals is binomially distributed[5], i.e.,

$$\mathsf{P}\{N_n(t) = k\} = \binom{n}{k} p^k (1-p)^{n-k}. \tag{2.2.1}$$

It is easy to show[6] that the expected number of arrivals during $[0,t]$ is $\mathsf{E}[N_n(t)] = \sum_{i=1}^n \mathsf{E}[B_i] = np$.

LET US TAKE the limit $n \to \infty$ and $p \to 0$, but such that the expected number of arrivals np during $[0,t]$ remains constant. Specifically, we take the limit such that $pn = \lambda t$, where the constant λ has the interpretation of the arrival rate[7] of customers. In this case,[8]

$$\lim_{\substack{n\to\infty, p\to 0\\ np=\lambda t}} \binom{n}{k} p^k (1-p)^{n-k} = e^{-\lambda t}\frac{(\lambda t)^k}{k!}. \tag{2.2.2}$$

We say that the random variable $N(t)$ associated with this distribution is Poisson distributed[9], i.e.,

$$\mathsf{P}\{N(t) = k\} = e^{-\lambda t}\frac{(\lambda t)^k}{k!}, \tag{2.2.3}$$

and we write $N(t) \sim P(\lambda t)$.[10] With $N(t)$, we define $N(s,t] = N(t) - N(s)$ as the number of customers arriving in the time period $(s,t]$.[11]

THE FAMILY of random variables $N_\lambda = \{N(t), t \geq 0\}$ is a Poisson process[12] with rate λ.[13] We remark that N_λ has stationary and independent increments[14]. Stationarity means that the distributions of the number of arrivals are the same for all intervals of equal length, that is, $N(s,t]$ has the same distribution as $N(u,v]$ if $t-s = v-u$. Independence means, roughly speaking, that knowing that $N(s,t] = n$, does not help to make any predictions about the value of $N(u,v]$ if the intervals $(s,t]$ and $(u,v]$ do not overlap.[15]

We next address a number of useful properties of the Poisson process. If $\Delta t \ll 1$ then for all t,[16]

$$\mathsf{P}\{N(t+\Delta t) = n \,|\, N(t) = n\} = 1 - \lambda\Delta t + o(\Delta t), \tag{2.2.4a}$$

$$\mathsf{P}\{N(t+\Delta t) = n+1 \,|\, N(t) = n\} = \lambda\Delta t + o(\Delta t), \tag{2.2.4b}$$

$$\mathsf{P}\{N(t+\Delta t) \geq n+2 \,|\, N(t) = n\} = o(\Delta t). \tag{2.2.4c}$$

[3] *Identically and independently distributed (i.i.d.)*

[4] *Bernoulli distributed*

[5] *Binomially distributed*

[6] [2.2.1]

[7] *Arrival rate*

[8] [2.2.2]

[9] *Poisson distributed*

[10] [2.2.3]

[11] *Note that $[0,t]$ is closed at both ends, but $(s,t]$ is open at the left.*

[12] *Poisson process*

[13] *A random process is a much more complicated mathematical object than a random variable: a process is a (possibly uncountable) set of random variables indexed by time, not* just *one random variable.*

[14] *Stationary and independent increments*

[15] *We refer to the literature on (mathematical) probability theory for further background.*

[16] [2.2.4]–[2.2.6]

The next equation says that if you know that an arrival occurred during $[0,t]$, the arrival is distributed uniformly on the interval $[0,t]$. If $s \in [0,t]$,[17]

$$\mathsf{P}\{N(s)=1 \,|\, N(t)=1\} = \frac{s}{t}. \tag{2.2.4d}$$

Note that this is independent of λ. Finally,[18]

$$\mathsf{E}[N(t)] = \lambda t, \tag{2.2.4e}$$
$$\mathsf{E}\left[(N(t))^2\right] = \lambda^2 t^2 + \lambda t, \tag{2.2.4f}$$
$$\mathsf{V}[N(t)] = \lambda t. \tag{2.2.4g}$$

[17] [2.2.10]

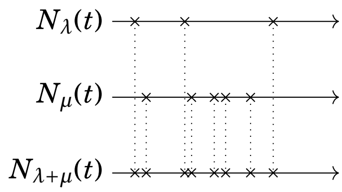
[18] [2.2.7]–[2.2.9]

THE RELATIVE VARIABILITY of service times is a very important concept in queueing theory. For instance, suppose the standard deviation of customer inter-arrival times is 1 minute. When the mean inter-arrival time is 1 hour, we are inclined to call the process regular, while if the mean is 1 minute, we would call it irregular. To differentiate between such cases, define the square coefficient of variation (SCV)[19] of a random variable X as

$$C^2 = \frac{\mathsf{V}[X]}{(\mathsf{E}[X])^2}. \tag{2.2.5}$$

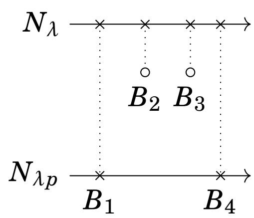
[19] *Square coefficient of variation (SCV)*

MERGING AND SPLITTING OF ARRIVAL PROCESSES often occurs in practice. Consider, for instance, the arrival processes N_λ of men and N_μ of women at a shop, see the figure at the right. Each cross represents an arrival; in the upper line it corresponds to a man, in the middle line to a woman, and in the lower line to an arrival of a general customer at the shop. Thus, the shop 'sees' the merged process of these two arrival processes. In fact, this merged process $N_{\lambda+\mu}$ is also a Poisson process[20] with rate $\lambda+\mu$.



[20] [2.2.12]

We can also *split*, or *thin*, a stream into several sub-streams. Model the stream of people passing by a shop as a Poisson process N_λ. In the figure at the right, we mark these arrivals as crosses at the upper line. With probability p a person decides, independent of anything else, to enter the shop; the crosses at the lower line are the customers that enter the shop. The Bernoulli random variable $B_1 = 1$ so that the first passerby enters the shop; the second passerby does not enter as $B_2 = 0$, and so on.



The concepts of merging and thinning are useful to analyze queueing networks, see Section 7.3. Suppose the departure stream of a machine splits into two sub-streams, e.g., a fraction p of the jobs moves on to another machine and the rest $(1-p)$ of the jobs leaves the system. Then we can model the arrival stream at the second machine as a thinned stream (with probability p) of the departures of the first machine. Merging occurs where the output streams of various stations arrive at another station.

WITH MOMENT-GENERATING FUNCTION we can simplify the derivations above; the last few exercises of this section show how to apply this.[21]

[21] *In general, it is hard to obtain closed-form expressions for the moment-generating function, but when it works, it is an easy and slick technique.*

Ex 2.2.1. Show that $\mathsf{E}[N_n(t)] = \sum_{i=1}^n \mathsf{E}[B_i] = np$.

Ex 2.2.2. Show (2.2.2)

Ex 2.2.3. What is the difference between $N_n(t)$ and $N(t)$?

Ex 2.2.4. Show (2.2.4a).

Ex 2.2.5. Show (2.2.4b).

Ex 2.2.6. Show (2.2.4c)

Ex 2.2.7. Show (2.2.4e).

Ex 2.2.8. Show (2.2.4f).

Ex 2.2.9. Show (2.2.4g).

Ex 2.2.10. Show (2.2.4d).

Ex 2.2.11. Show that the SCV of $N(t) \sim P(\lambda t)$ is equal to $1/(\lambda t)$. What does this mean for t large?

Ex 2.2.12. If the Poisson arrival processes N_λ and N_μ are independent, show with a conditioning argument that the merged process $N_\lambda + N_\mu$ is a Poisson process with rate $\lambda + \mu$.

Ex 2.2.13. If the Poisson arrival processes N_λ and N_μ are independent, show that

$$\mathsf{P}\left\{N_\lambda(t) = 1 \,\middle|\, N_\lambda(t) + N_\mu(t) = 1\right\} = \frac{\lambda}{\lambda + \mu}. \tag{2.2.6}$$

Note that the RHS does not depend on t.

Ex 2.2.14. Show with conditioning that thinning the Poisson process N_λ by means of Bernoulli random variables with success probability p results in a Poisson process $N_{\lambda p}$.

Ex 2.2.15. Show that $M_{N(t)}(s) = \exp(\lambda t(e^s - 1))$.

Ex 2.2.16. Use $M_{N(t)}$ to find $\mathsf{E}[N(t)]$ and $\mathsf{V}[N(t)]$.

Ex 2.2.17. Show with moment-generating functions that thinning the Poisson process N_λ by means of Bernoulli random variables with success probability p results in a Poisson process $N_{\lambda p}$.

Ex 2.2.18. If the Poisson arrival processes N_λ and N_μ are independent, use moment-generating functions to show that $N_\lambda + N_\mu$ is a Poisson process with rate $\lambda + \mu$.

Ex 2.2.19. Use moment-generating functions to prove (2.2.2).

2.3 QUEUEING PROCESS IN CONTINUOUS TIME

In Section 2.1 we modeled time as progressing in discrete chunks. However, we can also model queueing systems in continuous time, so that jobs can arrive at any moment in time and have arbitrary service times. In this section, we develop a set of recursions to construct the waiting times of jobs served in the sequence in which they arrive. First we concentrate on a situation in which there is one server available; at the

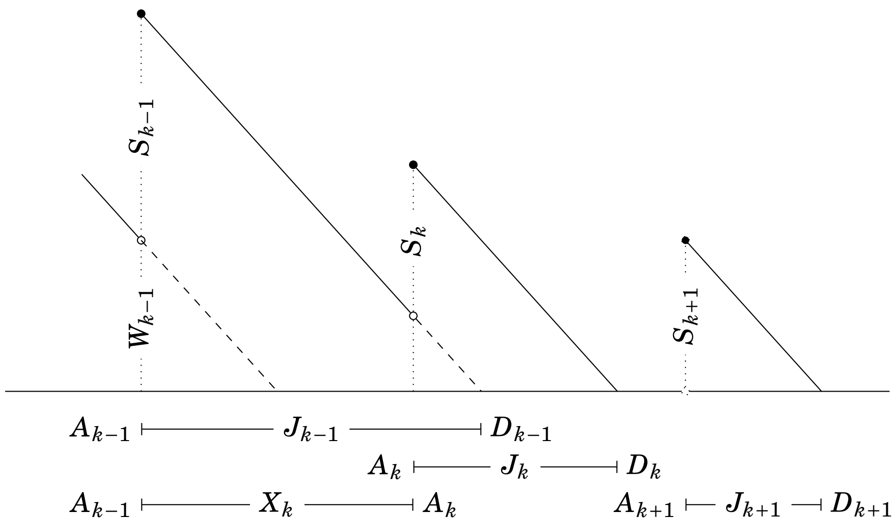


Figure 3: Construction of the single-server queue in continuous time. The virtual waiting time process is shown by the solid lines with slope -1.

end we extend this to multiple servers, and discuss some computational aspects.

Let's imagine that a machine starts working on a one-hour job at 9 am. When the next job arrives before 10 am, this second job has to wait in queue until the first job finishes at 10 am. Suppose next that the one-hour job arrives at 9 am but has to wait 5 hours before its production can start. When the next job arrives before $9+5+1=3$ pm, it has to wait, while if it arrives after 3 pm, it finds the machine free, and its service can start right away.

MORE GENERALLY, suppose we are given a sequence $\{X_k\}$ of *inter-arrival* times between jobs and a sequence $\{S_k\}$ of *service times*. When job $k-1$ has to wait a time W_{k-1} in queue, then adds its service time S_{k-1} to the waiting time, and a time X_k elapses between the arrival time of job $k-1$ and k, then job k has to wait in queue

$$W_k = [W_{k-1} + S_{k-1} - X_k]^+. \tag{2.3.1}$$

Observe[0] that with this recursion it is easy to compute the sequence of waiting times $\{W_k\}$ in queue: from the initial condition $W_0 = 0$ and $S_0 = 0$ we can obtain W_1, and then W_2, and so on, see Fig. 3.

[0] Note that we assume that job k 'reveals' its service time at the moment it arrives.

Henceforth, we always assume (implicitly) that the $\{X_k\}$ are i.i.d., the $\{S_k\}$ are i.i.d., and the $\{X_k\}$ are independent of the $\{S_k\}$.

The *sojourn time* J_k, is the time job k spends in the entire system. Thus,

$$J_k = W_k + S_k, \tag{2.3.2}$$

and a bit of thought will show that J_k can also be found from

$$J_k = [J_{k-1} - X_k]^+ + S_k. \tag{2.3.3}$$

IT IS CLEAR that we need a sequence $\{X_k\}$ of inter-arrival times, but such times are not always *measured*. However, if we know the number of arrivals $A(t)$ as a function of time t, we can reconstruct $\{X_k\}$. For instance, 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 $(s,t]$. Specifically, we define the arrival time[1] of job k as[2] [3]

$$A_k = \min\{t : A(t) \geq k\}, \quad A_0 = 0. \tag{2.3.4}$$

Once we have the sequence of arrival times $\{A_k\}$, the sequence of inter-arrival times[4] $\{X_k, k = 1,2,\dots\}$ between consecutive customers follows as

$$X_k = A_k - A_{k-1}. \tag{2.3.5}$$

Conversely, if the basic data consists of the inter-arrival times $\{X_k\}$, we find the arrival times with the recursion[5]

$$A_k = A_{k-1} + X_k, \quad A_0 = 0. \tag{2.3.6}$$

And then, from the arrival times $\{A_k\}$, we can define $A(t)$ as [6] [7]

$$A(t) = \max\{k : A_k \leq t\}. \tag{2.3.7}$$

Observe that the function $t \to A(t)$ is right-continuous.[8]

The virtual waiting time process[9] $\{V(t)\}$ is the amount of waiting when a job would arrive at time t. To construct[10] $\{V(t)\}$, we simply draw lines that start at points (A_k, W_k) and have slope -1, until the lines hit the x-axis, in which case the virtual waiting time remains zero until the next arrival occurs.

AFTER MOVING TO THE SERVER and completing its service, a job leaves the system. The departure time of the system[11] is job k is given by[12]

$$D_k = A_k + J_k. \tag{2.3.8}$$

With the sequence $\{D_k\}$, the number of departures $D(t)$ up to time t can be computed as[13]

$$D(t) = \max\{k : D_k \leq t\} = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \leq t}. \tag{2.3.9}$$

ONCE WE HAVE THE ARRIVAL and departure processes it is easy to compute the number of jobs in the system[14] at time t as, see Fig. 4,

$$L(t) = A(t) - D(t) + L(0), \tag{2.3.10}$$

where $L(0)$ is the number of jobs in the system at time $t = 0$; typically we assume that $L(0) = 0$.

In a queueing system, a job can be in queue or in service. We therefore distinguish between the number of jobs in the system $L(t)$, the number of jobs in queue $Q(t)$, and the number in service $L_s(t)$. Clearly, $L(t) = Q(t) + L_s(t)$.

It is important to realize that the queue length process $\{Q(t)\}$ at general time moments t can be quite different from the queue length process $\{Q(A_k-)\}$ as observed arriving jobs.[15]

[1] *Arrival time*

[2] *If we want to be mathematically precise, we must take* inf *rather than* min.

[3] *[2.3.2]*

[4] *Inter-arrival times*

[5] *[2.3.3]*

[6] *[2.3.4]*

[7] *For the mathematically inclined, we should consider $\{A(t,\omega), \omega \in \Omega\}$ as a counting process labeled by the samples ω in the sample space Ω, and not just look at one sample $A(t,\omega)$.*

[8] *You might want to prove this.*

[9] *Virtual waiting time process*

[10] *[2.3.9]*

[11] *Departure time of the system*

[12] *[2.3.5] and [2.3.7]*

[13] *[2.3.5]*

[14] *Number of jobs in the system*

[15] *Observe that we write A_k-, and not A_k; we need to be careful about left and right limits at jump epochs.*

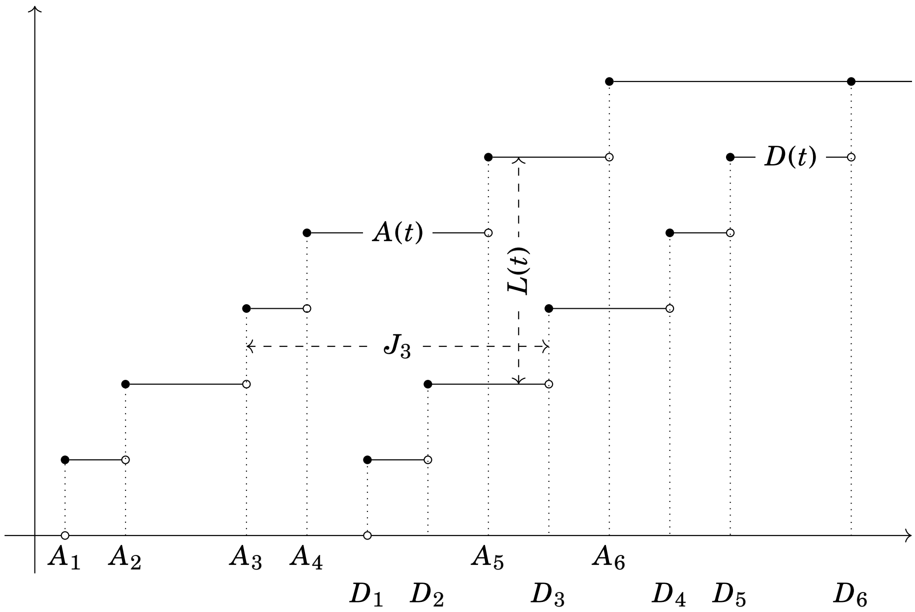


Figure 4: Relation between the arrival process $\{A(t)\}$, the departure process $\{D(t)\}$, the number in the system $\{L(t)\}$ and the sojourn times $\{J_k\}$.

LET US NEXT CONSTRUCT a multi-server FIFO queue in which the service of the first job in line starts when a server becomes free; if a server is free when a job arrives, the job's service starts right away.

Suppose there are m servers available, each with its own waiting line, like in a supermarket. When job k arrives, it sees a waiting time $w_{k,i}$ at line i. Of course, the job selects the line with the shortest waiting time,[16] which is $s_k = \arg\min\{w_{k,i} : i = 1,\dots,m\}$, and then it joins the end of that line.

[16] *This is not necessarily the same as the shortest queue.*

To formulate this as a recursion, we write $w_k = (w_{k,1},\dots,w_{k,m})$ for the vector of waiting times at the lines as seen by job k upon its arrival, e_i as the ith unit vector (a 1 at place i and zeros elsewhere), and $\mathbf{1} = (1,\dots,1)$. The waiting time of job k becomes $W_k = w_{k,s_k}$, and, in analogy with (2.3.1), the vector w_k updates as

$$w_{k+1} = [w_k + S_k e_{s_k} - X_{k+1}\mathbf{1}]^+, \tag{2.3.11}$$

where $[\cdot]^+$ applies element-wise.

It is useful to analyze the algorithmic complexity of this algorithm. For job k, we need to find the minimum in w_k, and compute and subtract $X_{k+1}\mathbf{1}$. The number of computational operations for this is m, as w_k and $\mathbf{1}$ contain m elements. For a simulation with N jobs, the total amount of operations is $N \times m$. However, by using a different implementation,[17] the complexity can be reduced to $N \times \log_2 m$, which is considerably faster when m and N are large.

[17] *With event stacks.*

Ex 2.3.1. Show that we can also define $A(t)$ as $A(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t}$.

Ex 2.3.2. Are the following mappings correct:

$$A_k : \mathbb{N} \to \mathbb{R}, \quad \text{job id (integer) to arrival time (real number),} \tag{2.3.12}$$

$$A(t) : \mathbb{R} \to \mathbb{N}, \quad \text{time (real number) to number of jobs (integer)?} \tag{2.3.13}$$

Ex 2.3.3. What is the meaning of $A_{A(t)}$, and of $A(A_n)$?

Ex 2.3.4. In view of the above, can $A(t)$ be defined as $\min\{k : A_k \geq t\}$ or as $\min\{k : A_k > t\}$?

Practice with definitions.

Ex 2.3.5. 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.

Ex 2.3.6. 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. Compute A_k, W_k, D_k as functions of k. Then find expressions for $A(t)$ and $D(t)$.

Ex 2.3.7. 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 $W_0 = 3$, what are the distributions of W_1 and W_2?

Ex 2.3.8. Explain the following recursions for a single server queue:

$$A_k = A_{k-1} + X_k, \quad D_k = \max\{A_k, D_{k-1}\} + S_k, \quad J_k = D_k - A_k. \tag{2.3.14}$$

Ex 2.3.9. Provide a specification of the virtual waiting time process $\{V(t)\}$ for all t.

Ex 2.3.10. In [2.3.6], find an expression for $L(A_k-)$.

Ex 2.3.11. Explain that if we know $\tilde{A}(t)$, i.e. the number of jobs that departed from the queue up to time t, then

$$Q(t) = A(t) - \tilde{A}(t), \qquad L_s(t) = \tilde{A}(t) - D(t) = L(t) - Q(t). \tag{2.3.15}$$

Ex 2.3.12. Consider a multi-server queue with m servers. Suppose that at some time t it happens that $\tilde{A}(t) - D(t) < m$, where $\tilde{A}(t)$ is the number of jobs that departed from the queue up to time t, but $A(t) - D(t) > m$. How can this occur?

Ex 2.3.13. Show that $L(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t < D_k}$ when the system starts empty.

Ex 2.3.14. Show that $L(A_k) > 0 \implies A_k \leq D_{k-1}$.

Ex 2.3.15. With the recursions (2.3.3) it is apparently easy to compute the waiting time (in queue), but it is less simple to compute the number of jobs in queue or in the system. In this exercise we develop an algorithm to compute the number of jobs in the system as seen by arrivals. Explain why the following (algorithmic efficient) procedure works:

$$L(A_k-) = L(A_{k-1}-) + 1 - \sum_{i=k-1-L(A_{k-1}-)}^{k-1} \mathbb{1}_{D_i < A_k}. \tag{2.3.16}$$

Ex 2.3.16. In [2.3.15], why do we take $i = k-1-L(A_{k-1}-)$ in the sum, and not $i = k-2-L(A_{k-1}-)$?

Ex 2.3.17. 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.

Ex 2.3.18. Implement the recursions for the multi-server queue in code, and run it on an example.

2.4 EXPONENTIAL DISTRIBUTION

In Section 2.2 we introduce the Poisson process to model the arrival process of jobs, and we use the Poisson distribution in simulations to generate the random number of jobs arriving in a period. Likewise, to model and simulate the continuous-time single-server queueing process of Section 2.3, we need to specify distributions for the inter-arrival times $\{X_k\}$ and service times $\{S_k\}$. In particular for the inter-arrival times the exponential distribution is useful as it is closely related to the Poisson distribution. In this section we concentrate on the properties of the exponential distribution; one of the most important is the memory-less property.

We say that a random variable X is exponentially distributed[0] with mean $1/\lambda$ if

$$\mathsf{P}\{X \leq t\} = 1 - e^{-\lambda t}, \tag{2.4.1}$$

and then we write $X \sim \text{Exp}(\lambda)$. The following properties hold:[1]

$$\mathsf{E}[X] = \lambda^{-1}, \tag{2.4.2a}$$
$$\mathsf{V}[X] = \lambda^{-2}, \tag{2.4.2b}$$
$$C^2 = 1, \tag{2.4.2c}$$
$$M_X(t) = \frac{\lambda}{\lambda - t}, \quad t < \lambda. \tag{2.4.2d}$$

WE NOW INTRODUCE another fundamental concept. A random variable X is called memoryless[2] when it satisfies

$$\mathsf{P}\{X > s + t \mid X > s\} = \mathsf{P}\{X > t\}. \tag{2.4.3}$$

In words, the probability that X is larger than some time $s+t$, conditional on it being larger than a time s, is equal to the probability that X is larger than t. Stated differently, even given that X did not occur before s, the probability that it takes t time units more to occur, is the same as if we did not have to wait for s time units to pass.

The remarkable fact is that an exponentially distributed random variable is memoryless[3]. The reverse—a continuous memoryless random variable is exponential—can also be proven, but that is harder[4].

The life span of human beings is not memoryless: take X as the life span of an arbitrary person born in 1880, and $s = t = 100$ years. Then $\mathsf{P}\{X > 100\}$ was small, but not zero, but $\mathsf{P}\{X > 200 \mid X > 100\} = 0$.[5] However, if X is the arrival time at the emergency room of a hospital of

[0] Exponentially distributed

[1] [2.4.1]–[2.4.4]

[2] Memoryless

[3] [2.4.7]

[4] Yushkevich and Dynkin [1969, Appendix 3]

[5] Even if you believe that Elvis Presley is still alive.

a patient with a broken arm. What can we say about X when we know that also an hour earlier a patient came in with broken arm? Not much, as most of us will agree.

The time scales we consider in queueing theory range from minutes to months.[6] On these timescales it often turns out that it is reasonable to model inter-arrival times of jobs as memoryless, hence exponentially distributed.

[6] *And of course there are exceptions.*

THE GEOMETRIC DISTRIBUTION is the discrete-time analog of exponential distribution. As such, geometric random variables are memoryless. To demonstrate this, consider a machine that produces items. An item is correct with probability p, and faulty with probability $1-p$, independent of the correctness of any other item. Let X be the number of items produced until a failure occurs. Then $\mathsf{P}\{X=m\} = p^{m-1}(1-p)$, from which easily follows that

$$\mathsf{P}\{X > n+m \,|\, X > m\} = \frac{\mathsf{P}\{X > n+m\}}{\mathsf{P}\{X > m\}} = \frac{p^{n+m}}{p^m} = \mathsf{P}\{X > n\}. \tag{2.4.4}$$

THERE IS A CLOSE link between the Poisson process N and the exponential distribution. In fact, 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. whose common distribution is that of the random variable $X \sim \text{Exp}(\lambda)$.[7,8]

[7] *By this we mean that there is a random variable X such that $\mathsf{P}\{X_k \leq x\} = \mathsf{P}\{X \leq x\}$ for all k.*

[8] *[2.4.8]–[2.4.12].*

Thus, if you find it reasonable to model inter-arrival times as memoryless, then the number of arrivals in an interval is necessarily Poisson distributed. And, if you find it reasonable that the occurrence of an event in a small time interval is constant over time and independent from one interval to another, then the arrival process is Poisson, and the inter-arrival times are exponential.

Ex 2.4.1. Show (2.4.2a).

Ex 2.4.2. If $X \sim \text{Exp}(\lambda)$, show that $\mathsf{E}\left[X^2\right] = \frac{2}{\lambda^2}$.

A useful intermediate step to compute the variance.

Ex 2.4.3. Show (2.4.2b).

Ex 2.4.4. Show (2.4.2d).

Ex 2.4.5. Use $M_X(t)$ to show (2.4.2a) and (2.4.2b).

Ex 2.4.6. Show (2.4.2c).

Ex 2.4.7. Show that $X \sim \text{Exp}(\lambda)$ is memoryless.

Ex 2.4.8. If N_λ is a Poisson process with rate λ, show that the time X_1 to the first arriving job is $\text{Exp}(\lambda)$.

Poisson $\Longrightarrow$ Exponential.

Ex 2.4.9. Assume that inter-arrival times $\{X_i\}$ are i.i.d. $\sim \text{Exp}(\lambda)$. Let the arrival time of the ith job be $A_i = \sum_{k=1}^{i} X_k$. Show that $\mathsf{E}[A_i] = i/\lambda$.

In this and the next exercises we show that exponential $\Longrightarrow$ Poisson.

Ex 2.4.10. Prove that A_i has density $f_{A_i}(t) = \lambda e^{-\lambda t} \frac{(\lambda t)^{i-1}}{(i-1)!}$.

Continuation of [2.4.9].

Ex 2.4.11. Use f_{A_i} of [2.4.10] to show that $\mathsf{E}[A_i] = i/\lambda$.

Continuation of [2.4.10].

Ex 2.4.12. If the inter-arrival times $\{X_i\}$ are i.i.d. $\sim \text{Exp}(\lambda)$, prove that the number $N(t)$ of arrivals during the interval $[0,t]$ is Poisson distributed.

Continuation of [2.4.10].

Ex 2.4.13. If $X \sim \text{Exp}(\lambda), S \sim \text{Exp}(\mu)$ and independent, show that $Z = \min\{X,S\} \sim \text{Exp}(\lambda+\mu)$, hence $\mathsf{E}[Z] = (\lambda+\mu)^{-1}$.

This result can be anticipated when you think about merging Poisson processes.

Ex 2.4.14. If $X \sim \text{Exp}(\lambda)$, $S \sim \text{Exp}(\mu)$ and independent, show that

Now think about splitting Poisson processes.

$$\mathsf{P}\{X \leq S\} = \frac{\lambda}{\lambda+\mu}. \tag{2.4.5}$$

3 FROM TRANSIENT TO STEADY-STATE ANALYSIS

With the tools developed in Chapter 2 we can simulate queueing processes. In this chapter we make a start with developing mathematical models of queueing systems. However, as we will see in Section 3.2, the mathematical characterization of the transient behavior of even simple queueing system is already extremely complicated. Thus, we have to lower our goals, and for this reason we will focus on the steady-state behavior of queueing systems.

We introduce the concepts of stability and load in Section 3.3 and express these in terms of the arrival and service rate. The notions of arrival and service rate are crucial because they capture our intuition that when jobs can be served faster than they arrive, on average, the queue does not systematically drift to infinity. Once this is ensured, we can properly define a number of measures to characterize the performance of the queueing system, such as the average waiting time, see Section 3.4.

Before introducing these definitions, however, we introduce in Section 3.1 some notational shorthands to characterize the type of queueing process. We provide in Section 3.5 an overview of the relations introduced in this chapter.

3.1 KENDALL'S NOTATION

As became apparent in Sections 2.1 and 2.3, the construction of a single-station queueing process involves three main elements: the distribution of job inter-arrival times and the service times, and the number of servers present to process jobs.

To characterize the type of queueing process it is common to use Kendall's abbreviation[0] $A/B/c/K$, where A is the distribution of the inter-arrival times, B the distribution of the service times, c the number of servers, and K the system size, i.e., the total number of customers that can be simultaneously present, whether in queue or in service.[1] In this notation it is assumed that jobs are served in FIFO order, and, in this book, we only consider the FIFO service discipline.

[0] *Kendall's abbreviation*

[1] *The meaning of K differs among authors. Sometimes it stands for the capacity of the queue, not the entire system. In this book K corresponds to the system's size.*

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

A FEW IMPORTANT EXAMPLES are the following queueing processes: $M/M/1$, $M/G/1$ and $G/G/c$. A model that is often used to determine the number of beds needed in (a ward of) a hospital is the $M/M/c/c/c$ queue. The motivation is as follows. Practice tells us that patient inter-arrival times are memoryless, hence exponentially distributed. Data of patients treatment times shows that these times are also well-described by an exponential distribution. Next, there are c beds available, and each bed can serve one patient. When all beds are occupied, the hospital is 'full'.

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

3.2 QUEUEING PROCESSES AS REGULATED RANDOM WALKS

In this section, we provide an elegant construction of a queueing process based on a *random walk*. This serves two goals. The fist is to show that queueing theory is essentially based on concepts of fundamental interest in probability theory (the random walk), hence is strongly related to many other applications of random walks, such as finance, inventory theory, and insurance theory. The second is to show that it is (very) hard to characterize the transient behavior of a queueing process. Thus, in the rest of the book we will only study queueing systems in steady-state.

In the construction of queueing processes as set out in Section 2.1 we are given two sequences of i.i.d. random variables: the number of arrivals $\{a_k\}$ per period and the service capacities $\{c_k\}$, see (2.1.1). Observe that in (2.1.1) the process $\{L_k\}$ shares a resemblance to a random walk $\{Z_k, k = 0, 1, \ldots\}$ with Z_k given by

$$Z_k = Z_{k-1} + a_k - c_k. \tag{3.2.1}$$

To see that $\{Z_k\}$ is indeed a random walk, observe that Z makes jumps of size $a_k - c_k, k = 1, \ldots$, 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.[0]

[0] $\{Z_k\}$ is 'free', i.e., it can take positive and negative values, whereas a queue length process is restricted to the non-negative integers.

It is interesting to express the (discrete-time) queueing process $\{L_k\}$ in terms of $\{Z_k\}$. From (3.2.1), it is evident that $a_k - c_k = Z_k - Z_{k-1}$, hence,

$$L_k = [L_{k-1} + a_k - d_{k-1}]^+ = [L_{k-1} + Z_k - Z_{k-1}]^+. \tag{3.2.2}$$

It follows that $L_k - Z_k = \max\{L_{k-1} - Z_{k-1}, -Z_k\}$, and, by completing the recursion,[1] we obtain

$$L_k = Z_k - \min_{1 \le i \le k} Z_i \wedge 0, \tag{3.2.3}$$

where $a \wedge b = \min\{a, b\}$.

[1] [3.2.1]

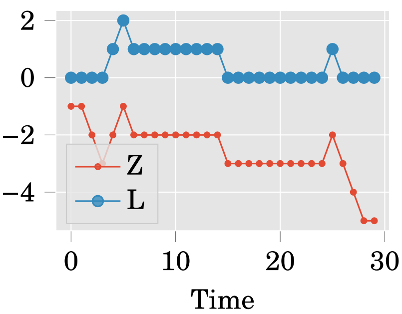


Figure 5: An instance of (3.2.1).

This recursion for L_k leads to nice graphs. In Fig. 5 we take $a_k \sim B(0.3)$ and $c_k \sim B(0.4)$. In Fig. 6, $a_k \sim B(0.49)$ the random walk behaves according to

$$Z_k = Z_{k-1} + 2a_k - 1. \tag{3.2.4}$$

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

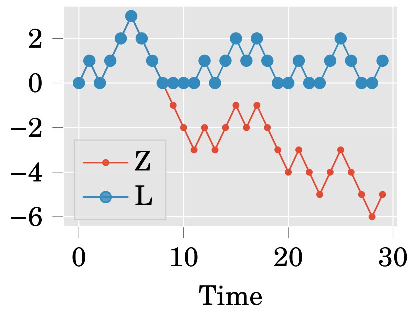


Figure 6: An instance of (3.2.4).

WHAT CAN WE SAY ABOUT the transient (time-dependent) behavior of $\{L_k\}$ with the above? To obtain insight into this question, let us suppose that $a_k \sim P(\lambda)$ and $c_k \sim P(\mu)$. Then, by [2.2.12],

$$Z_k = Z_0 + N_{\lambda k} - N_{\mu k}. \tag{3.2.5}$$

With a bit of work, we can show[2] that

[2] [3.2.2]

$$\mathsf{P}\{Z_k = n\} = e^{-(\lambda+\mu)k}(\lambda k)^{n-m} \sum_{j=0}^{\infty} \frac{(\lambda\mu k^2)^j}{j!(n-m+j)!}. \tag{3.2.6}$$

We can try to combine this expression with (3.2.3) to characterize the transient distribution of $\{L_k\}$, it is apparent that this will not lead to any simple function.

As we will see later, the $M/M/1$ queue is about the simplest queueing system to analyze; other queueing systems are (much) more complicated. As we already have to give up our attempts to analyze the transient $M/M/1$ queue, we do not pursue this topic any further and contend ourselves henceforth with the analysis of queueing systems in the limit as $t \to \infty$.

Ex 3.2.1. Show that L_k satisfies the relation $L_k = Z_k - \min_{1\leq i\leq k} Z_i \wedge 0$.

Ex 3.2.2. Show that when $n > m$ and $Z_0 = m$,

Note that the difference of two Poisson random variables is not Poisson, in contrast to the sum.

$$\mathsf{P}\{Z_k = n\} = e^{-(\lambda+\mu)k}(\lambda k)^{n-m} \sum_{j=0}^{\infty} \frac{(\lambda\mu k^2)^j}{j!(n-m+j)!}. \tag{3.2.7}$$

Ex 3.2.3. Suppose for the $G/G/1$ that a job sees n jobs in the system upon arrival. Use the central limit theorem to estimate the distribution of the waiting time in queue for this job.

When we start with a really large queue, we characterize the time to clear the system in Section 7.1.

3.3 RATE, STABILITY AND LOAD

In this section, we develop a number of measures to characterize the performance of queueing systems in steady-state. In particular, we define the load, which is, arguably, the most important performance measure of a queueing system to check.

WE FIRST FORMALIZE the arrival rate and departure rate in terms of the arrival and departure processes $\{A(t)\}$ and $\{D(t)\}$, see Section 2.3. The arrival rate[0] is the long-run average number of jobs that arrive per unit time along a sample path, i.e.,[1]

[0] *Arrival rate*

[1] *This limit does not necessarily exist if $A(t)$ is some pathological function.*

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t}. \tag{3.3.1}$$

Likewise, define the departure rate[2] as

[2] *Departure rate*

$$\delta = \lim_{t\to\infty} \frac{D(t)}{t}. \tag{3.3.2}$$

Henceforth we assume that both limits are finite.

OBSERVE THAT, if the system is empty[3] at time 0, the number of departures must be smaller than or equal to the number of arrivals, i.e., $D(t) \le A(t)$ for all t. Therefore,

$$\delta = \lim_{t\to\infty} \frac{D(t)}{t} \le \lim_{t\to\infty} \frac{A(t)}{t} = \lambda. \tag{3.3.3}$$

[3] *Why is this necessary to require?*

It is intuitively obvious that when $\lambda > \delta$, the system length process $L(t) = L(0) + A(t) - D(t) \to \infty$ as $t \to \infty$. We therefore call a system rate-stable[4] if

$$\lambda = \delta. \tag{3.3.4}$$

[4] *Rate-stable*

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

SUPPOSE THAT WE are given a sequence of i.i.d. inter-arrival times $\{X_k\}$ whose common distribution is that of the random variable X. Then we can relate the arrival rate λ to the mean inter-arrival time $\mathsf{E}[X]$ as follows.[5] Observe that at time $t = A_n$ precisely n arrivals occurred. Then, with [2.3.3] we see that $A(A_n) = n$, and therefore,

[5] *The existence of the limit (3.3.1) is also guaranteed under this assumption.*

$$\frac{1}{n}\sum_{k=1}^{n} X_k = \frac{A_n}{n} = \frac{A_n}{A(A_n)}. \tag{3.3.5}$$

But since $A_n \to \infty$ if $n \to \infty$, it follows from (3.3.1) that the average inter-arrival time between two consecutive jobs is

$$\mathsf{E}[X] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \lim_{n\to\infty} \frac{A_n}{A(A_n)} = \lim_{t\to\infty} \frac{t}{A(t)} = \frac{1}{\lambda}, \tag{3.3.6}$$

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

IN (3.3.6) WE REPLACED the limit with respect to n by a limit with respect to t. To show that this is allowed, observe 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. Therefore,

$$A_{A(t)} \le t < A_{A(t)+1} \Leftrightarrow \frac{A_{A(t)}}{A(t)} \le \frac{t}{A(t)} < \frac{A_{A(t)+1}}{A(t)} = \frac{A_{A(t)+1}}{A(t)+1}\frac{A(t)+1}{A(t)}. \tag{3.3.7}$$

Now $A(t)$ is a counting process such that $A(t) \to \infty$ as $t \to \infty$, therefore,

$$\lim_{n\to\infty} \frac{A_n}{n} = \lim_{t\to\infty} \frac{A_{A(t)}}{A(t)} = \lim_{t\to\infty} \frac{A_{A(t)+1}}{A(t)+1}, \tag{3.3.8}$$

where the third limit follows trivially from the second. Finally, because $(A(t)+1)/A(t) \to 1$, we arrive at the equality $\lim_{t\to\infty} t/A(t) = \lim_{n\to\infty} A_n/n$.

CONSIDER THE $G/G/1$ QUEUE.[6] Let S_k be the required service time of the kth job to be served, so that $U_n = \sum_{k=1}^{n} S_k$ is the total service time of the first n jobs. Letting $U(t) = \max\{n : U_n \le t\}$, we define the service rate[7] or processing rate[8] as

[6] *See [3.3.4] for an extension to G/G/c queues.*

[7] *Service rate*

[8] *Processing rate*

$$\mu = \lim_{t\to\infty} \frac{U(t)}{t}. \tag{3.3.9}$$

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

$$\mathsf{E}[S] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} S_k = \lim_{n\to\infty} \frac{U_n}{n} = \lim_{n\to\infty} \frac{U_n}{U(U_n)} = \lim_{t\to\infty} \frac{t}{U(t)} = \frac{1}{\mu}. \tag{3.3.10}$$

ONCE WE HAVE the arrival and service rate, we define the load[9] as the rate at which work arrives:

$$\text{Load } = \frac{\lambda}{\mu} = \lambda\,\mathsf{E}[S] = \frac{\mathsf{E}[S]}{\mathsf{E}[X]}. \tag{3.3.11}$$

For the $G/G/c$ queue, we define the utilization[10] as,[11]

$$\text{Utilization } = \rho = \lambda\,\mathsf{E}[S]/c. \tag{3.3.12}$$

It is easy to check with a simulation of the $G/G/1$ queue that $L(t)$ increases at rate $\lambda - \mu$ when $\lambda > \mu$, while $L(t) \approx L(0) + (\lambda-\mu)t$ when $\lambda < \mu$ and $L(0)$ large, until the system is empty.[12]

For this reason the utilization is, perhaps, the most important performance measure of a queueing system to check: when $\rho \geq 1$, we are 'in trouble', when $\rho < 1$, we are 'safe'. We therefore require in the sequel that $\rho < 1$.

[9] *Load*

[10] *Utilization*

[11] *Pay attention, only for the $G/G/1$ queue, the load is equal to the utilization.*

[12] *It turns out that, when $\mathsf{E}[X] = \mathsf{E}[S]$ but $\mathsf{V}[X-S] > 0$, the queue length process behaves in a very peculiar way. This is due to the fact that the symmetric random walk has some unexpected behavior, and Section 3.2 shows that queueing systems and random walks are intimately related.*

Ex 3.3.1. Can you make an arrival process such that $A(t)/t$ does not have a limit?

Ex 3.3.2. If the system starts empty, then we know that the number $L(t)$ in the system at time t is equal to $A(t) - D(t)$. Show that the system is rate-stable if $L(t)$ remains finite, or, more generally, $L(t)/t \to 0$ as $t \to \infty$.

Ex 3.3.3. Show that $\mathsf{E}[X_k - S_k] < 0$ implies that $\rho < 1$.

Ex 3.3.4. Consider a queueing system with c servers with identical production rates μ. What would be a reasonable stability criterion for this system?

3.4 (LIMITS OF) EMPIRICAL PERFORMANCE MEASURES

In Section 2.3 we used the arrival process $\{A(t)\}$ and the service times $\{S_k\}$ to construct the waiting times $\{W_k\}$, sojourn times $\{J_k\}$, and the queueing process $\{L(t)\}$. If the queueing system is rate-stable, we can sensibly define several long-run average performance measures.

DEFINE THE expected waiting time in queue[0] as

$$\mathsf{E}[W] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} W_k. \tag{3.4.1}$$

Note that this is the limit of waiting times $\{W_k\}$ as *observed by arriving jobs*: the first job has to wait W_1 in queue, the second W_2, and so on.[1] The expected sojourn time[2] is defined likewise. The *distribution* of the waiting time as seen by arrivals can be found by counting:

$$\mathsf{P}\{W \leq x\} = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{W_k \leq x}, \tag{3.4.2}$$

[0] *Expected waiting time in queue*

[1] *We colloquially say that a statistic based on the sampling of arriving jobs is 'as seen by arrivals'.*

[2] *Expected sojourn time*

and for the distribution of J a similar definition applies. The average number of jobs[3] in the system is given by

$$\mathsf{E}[L] = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} L(A_k-), \tag{3.4.3}$$

since $L(A_k-)$ is the number of jobs in the system at the arrival epoch of the kth job. The distribution of L follows also from counting, compare (3.4.2).

[3] *Average number of jobs*

A RELATED SET OF performance measures follows by tracking the system's behavior over time and taking the *time-average*.[4]

Assuming the limit exists, we use (2.3.10) to define the time-average number of jobs[5] as [6]

$$\mathsf{E}[L] = \lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)\,\mathrm{d}s. \tag{3.4.4}$$

The *time-average fraction of time the system contains at most m jobs* is defined as

$$\mathsf{P}\{L \leq m\} = \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathbb{1}_{L(s)\leq m}\,\mathrm{d}s. \tag{3.4.5}$$

[4] *Now we say that such performance measures are as 'seen by the server'.*

[5] *Time-average number of jobs*

[6] *Even though the symbols are the same, this expectation is not necessarily the same as (3.4.3).*

PROVING THE EXISTENCE of the above limits requires a considerable amount of mathematics.[7] Here we sidestep all such fundamental issues, and simply assume that in the examples we consider all is well. The limiting random variables are known as the steady-state[8] or stationary[9] limits.

[7] *See Asmussen [2003] if you are interested.*

[8] *Steady-state*

[9] *Stationary*

BESIDES THAT THE transient analysis of queueing systems is difficult,[10] there is another reason why most queueing theory is concerned with the analysis of the system in steady state: Often the system convergences rapidly to the steady-state situation.[11] Hence, for most practical purposes, the knowledge of the steady-state performance measures suffices. Let us provide some intuition for this phenomenon by means of an example.

[10] *See Section 3.2*

[11] *In some technical sense, see Asmussen [2003].*

Suppose that X_k is uniformly distributed on the set $\{1,2,4\}$ and S_k uniform on the set $\{1,2,3\}$.[12] Starting with $W_0 = 5$, we like to construct the *distribution* of waiting times with the rule $W_k = [W_{k-1} + S_{k-1} - X_k]^+$. Observe that this rule contains three steps. First, we compute a new random variable $Z_k = W_{k-1} + S_{k-1}$, then $Z_k' = Z_k - X_k$, and finally $[Z_k']^+$. In other words, we first compute the sum of two independent random variables, then the difference, and finally apply a function $[.]^+$.

[12] *Hence, $\rho < 1$.*

Now, when X and Y are independent with densities f_X and f_Y, then it is well-known that

$$f_{X\pm Y}(n) = \sum_i \sum_j f_X(i) f_Y(j) \mathbb{1}_{i\pm j=n}, \quad f_{h(X)}(n) = \sum_i f_X(i) \mathbb{1}_{h(i)=n}. \tag{3.4.6}$$

We implement these rules in the code below, but slightly more efficiently.[13] For instance, the computation of the distribution of $X+Y$, $X-Y$, and so on, have much in common. We therefore build one method `apply_operator` that applies an operator like $+$ or $-$ to two independent random variables. Then we use this code to compute the sum in line 24, and the difference in line 27.

[13] *We advice the reader to study this code well; there is much to learn.*

```python
from collections import defaultdict
import operator

class RV(defaultdict):
    def __init__(self, p=None):
        super().__init__(float)
        if p:
            for (i, pi,) in p.items():
                self[i] = pi

    def apply_operator(self, Y, op):
        R = RV()
        for (i, pi,) in self.items():
            for (j, pj,) in Y.items():
                R[op(i, j)] += pi * pj
        return R

    def apply_function(self, h):
        R = RV()
        for (i, pi,) in self.items():
            R[h(i)] += pi
        return R

    def __add__(self, X):
        return self.apply_operator(X, operator.add)

    def __sub__(self, X):
        return self.apply_operator(X, operator.sub)

    def plus(self):
        return self.apply_function(lambda x: max(x, 0))

    def support(self):
        return sorted(self.keys())

    def pmf(self):
        return [self[k] for k in self.support()]
```

WE ARE NOW ready to compute and plot the pmf of W_k for increasing values of k.

```python
from matplotlib.pylab import plt
import tikzplotlib
from matplotlib import style
style.use('ggplot')

W = RV({5: 1})
X = RV({1: 1 / 3, 2: 1 / 3, 4: 1 / 3,})
```

```python
S = RV({1: 1 / 3, 2: 1 / 3, 3: 1 / 3,})

for n in range(1, 21):
    W += S - X
    W = W.plus()
    if n % 5 == 0:
        plt.plot(W.support(), W.pmf(), label="k={}".format(n))
```

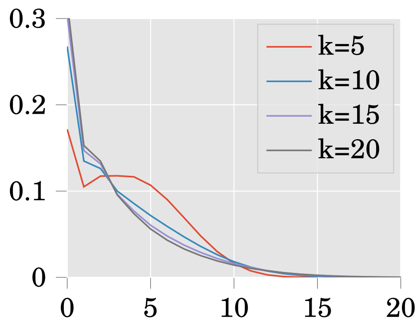


Figure 7: The pmf of W_k.

With the code above we make Fig. 7. Clearly, for $k = 5$, the pmf still contains a 'hump' just below $x = 5$, which is due the starting value of $W_0 = 5$. However, for $k \geq 10$, the pmf hardly changes. The sequence of pmfs seems to converge rather fast as k increases.

As an aside for the python aficionados, the code can be made yet a bit cleaner with defining `plus = operator.methodcaller("plus")` so that we can write `W = plus(W + S - X)`.

Ex 3.4.1. Design a queueing system to show that the average number of jobs in the system as seen by the server can be very different from what customers see upon arrival.

Ex 3.4.2. If $L(t)/t \to 0$ as $t \to \infty$, can it still be true that $\mathsf{E}[L] > 0$?

Ex 3.4.3. Consider the discrete-time model specified by (2.1.1). Provide an algorithm to estimate $\mathsf{P}\{L \leq m\}$ with simulation of a queueing situation in which the first job of the batch sees $L_{k-1} - d_k$ jobs in the system upon arrival, the second sees $L_{k-1} - d_k + 1$ jobs, and so on.

Performance measures obtained by sampling in discrete-time queueing models require some extra attention.

3.5 GRAPHICAL SUMMARY

Here is, in graphical form, an overview to show the relation between the concepts developed in this chapter.

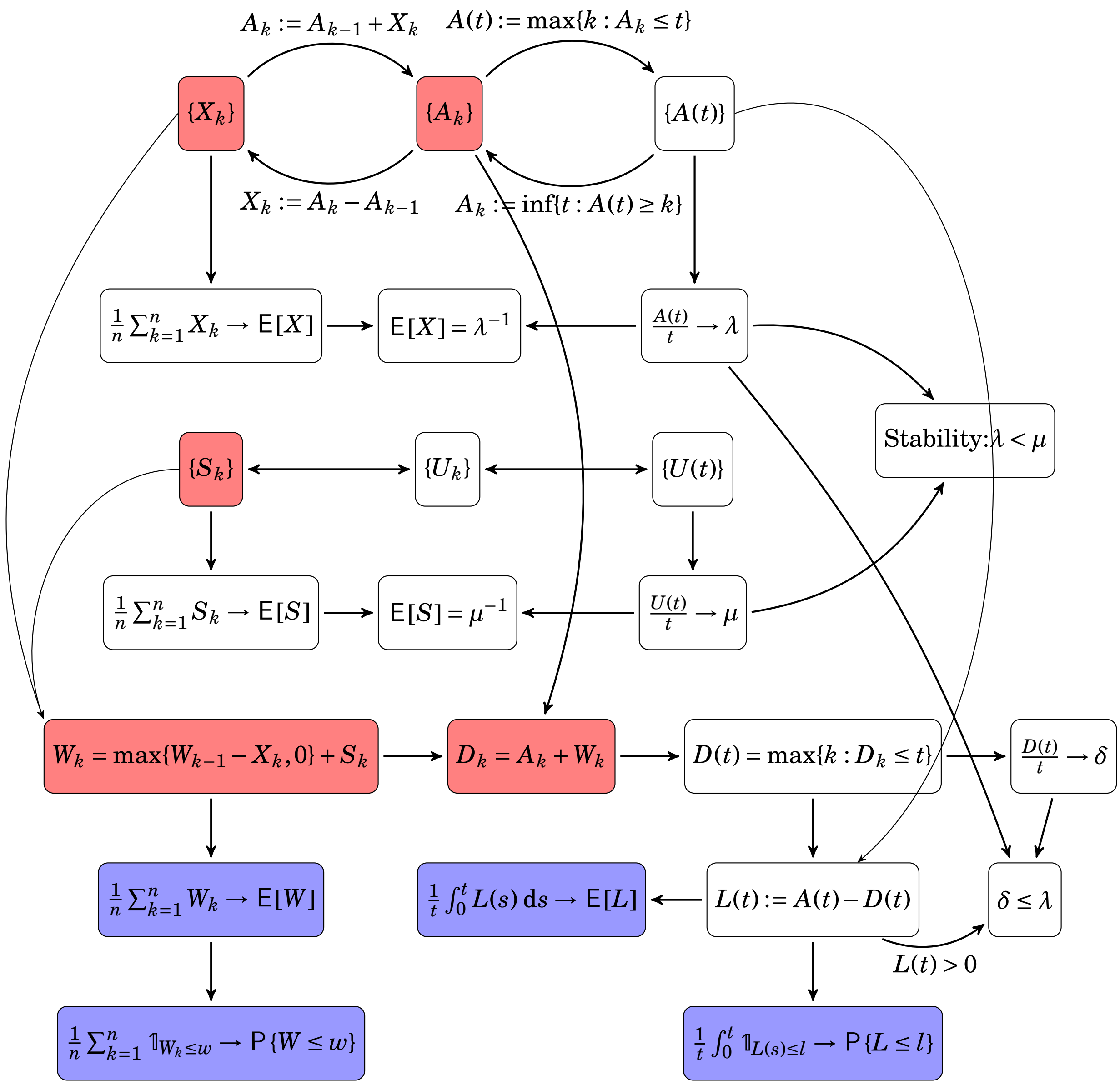


Figure 8: Here we sketch the relations between the construction of the $G/G/1$ queue from the primary data, i.e., the inter-arrival times $\{X_k; k \geq 0\}$ and the service times $\{S_k; k \geq 0\}$, and different performance measures.

4 APPROXIMATE QUEUEING MODELS

In the previous two chapters we learned how to construct and simulate queueing processes. Simulation is a powerful tool but one of its limitations is that it does not easily provide insight into structural behavior of systems. For this we need theoretical models, and the derivation of such models form the contents of the remainder of the book.

In this chapter we discuss two formulas that might be considered as the most important formulas to understand the behavior of queueing systems. The first is Sakasegawa's formula that approximates the expected queueing time in a $G/G/c$; the second characterizes the propagation of variability through a tandem network of $G/G/c$ queues. With a bit of exaggeration, it is justified to say that the entire philosophy behind lean manufacturing and the world-famous Toyota production system are based on the principles that can be derived from these two formulas.

Here we take these formulas for granted, but focus on the insights they provide into the performance of queueing systems and how to use them to guide improvement procedures for production and service systems. In Section 6.4 we provide the theoretical background of Sakasegawa's formula.

In Section 4.1 we introduce Sakasegawa's formula and discuss the main insights it offers. Then we illustrate how to use this formula to estimate waiting times in three queueing settings in which the service process is interrupted. In the first case, Section 4.2, the server has to produce jobs from different families, and there is a change-over time required to switch from one production family to another. As such setups reduce the time the server is available, the load must increase. In fact, to reduce the load, the server produces in batches of fixed sizes. In the second case, in Section 4.3, the server sometimes requires small adjustments, for instance, to prevent the production quality to degrade below a certain level. Clearly, such adjustments are typically not required during a job's service; however, they can occur between any two jobs. As a consequence, the number of jobs served between two such adjustments (or setups) is not constant, hence different from batch production where batch sizes are constant. In the third example, in Section 4.4, quality problems or break downs can occur during a job's service. These makes job service times more variable, which leads to longer expected queueing times. In the final Section 4.5, we concentrate on tandem queues.

In passing, we use some interesting results of probability theory and the Poisson process, which we use again in, for instance, Chapter 7.

4.1 $G/G/c$ QUEUE: SAKASEGAWA'S FORMULA

In this section, we discuss Sakasegawa's formula by which we can estimate the expected waiting time in queue for the $G/G/c$ queue. In the exercises we show how to use this formula, in Section 6.4 we provide the theoretical underpinning.

While there is no expression available to compute the exact expected waiting time for the $G/G/c$ queue, Sakasegawa's formula provides a reasonable approximation. This takes the form

$$\mathsf{E}[W] = \frac{C_a^2 + C_s^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{1-\rho} \frac{\mathsf{E}[S]}{c}, \tag{4.1.1a}$$

where $C_a^2 = \mathsf{V}[X]/(\mathsf{E}[X])^2$ is the SCV of the inter-arrival times, $C_s^2 = \mathsf{V}[S]/(\mathsf{E}[S])^2$ is the SCV of the service times, and the *utilization* of the station[0] is given by

$$\rho = \frac{\lambda \mathsf{E}[S]}{c} \tag{4.1.1b}$$

where λ is the rate at which jobs arrive at the system, $\mathsf{E}[S]$ is the expected service time, and c the number of servers.

[0] Here is a subtle point. When there are multiple machines, the utilization of each machine need not be equal to ρ. For instance, if there is a preference to choose the 'left-most' machine whenever it is free, then the utilization of this machine is larger than ρ. Only when the machines have the same speed, and the routing *is such that each machine receives a fraction λ/c of jobs, the machines have the same utilization.*

IT IS CRUCIAL TO MEMORIZE the insights into the performance of queueing systems that this formula offers. Even though (4.1.1a) is an approximation, it proves to be exceedingly useful when designing queueing systems and analyzing the effect of certain changes.

First, we see that $\mathsf{E}[W] \sim (1-\rho)^{-1}$. Consequently, when ρ is large, the waiting time is (very) large. And, not only is $\mathsf{E}[W]$ large, it is also extremely *sensitive* to the actual value of ρ. Clearly, such situations must be avoided, and therefore, when trying to improve a queueing system, the first focus should be on reducing ρ.

Second, $\rho = \lambda \mathsf{E}[S]/c$. Thus, when ρ must be made smaller, we have only three options.[1] We can reduce the arrival rate λ of jobs, for instance by blocking demand, or sending it elsewhere such as to another machine.[2] We can make $\mathsf{E}[S]$ smaller by replacing a server with a faster one or by shifting some of the processing steps of a job to other servers, thereby making job sizes smaller. Finally, we can add servers, which reduce ρ quite significantly when c is small.[3] If technically possible, adapting c is a very effective mechanism to control waiting times.

[1] And not more!

[2] And if customers have a choice, they will take their own measures, simply by going elsewhere.

[3] This is precisely what you see in supermarkets.

Third, $\mathsf{E}[W] \sim C_a^2$ and $\mathsf{E}[W] \sim C_s^2$, which implies that when job inter-arrival or service times are very variable, $\mathsf{E}[W]$ is large. Thus, after ensuring that ρ is sufficiently small, it becomes important to concentrate on reducing on C_a^2 and C_s^2.[4]

[4] It works the other way too: systems with low variability can deal with higher load.

Finally, $\mathsf{E}[W] \sim \mathsf{E}[S]/c$. This says that, from the perspective of a job in queue, average job service times are c times as short as 'its own service time'.

CLEARLY, SAKASEGAWA'S EQUATION requires an estimate of C_a^2 and C_s^2. Now it is not always easy in practice to determine the actual service time distribution, one reason being that service times are often only estimated by a planner, but not actually measured. Similarly, the actual arrival moments of jobs are often not registered, only just the date, or perhaps the hour, that a customer arrived. Hence, it is often not possible to directly estimate C_a^2 and C_s^2 from the information that is available.

However, when the number of arrivals per period $\{a_n\}$ has been logged for some time, we can use $\{a_n\}$ to estimate C_a^2 as[5]

$$C_a^2 \approx \frac{\tilde{\sigma}^2}{\tilde{\lambda}}, \tag{4.1.2}$$

where

$$\tilde{\lambda} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n a_i, \quad \tilde{\sigma}^2 = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n a_i^2 - \tilde{\lambda}^2. \tag{4.1.3}$$

We derive this results in steps.[6]

First recall some results of earlier sections. Let the inter-arrival time X have mean $1/\lambda$ and variance σ^2, so that

$$C_a^2 = \frac{\mathsf{V}[X_i]}{(\mathsf{E}[X_i])^2} = \frac{\sigma^2}{1/\lambda^2} = \lambda^2\sigma^2. \tag{4.1.4}$$

Let A_k be the arrival time of the kth arrival, see (2.3.6), and $A(t)$ the number of arrivals up to time t, see (2.3.7). Consider the following useful relation between $A(t)$ and A_k, see [2.4.10],

$$\mathsf{P}\{A(t) < k\} = \mathsf{P}\{A_k > t\}. \tag{4.1.5}$$

Since the inter-arrival times have finite mean and second moment by assumption, we use [3.2.3] to see that

$$\lim_{k\to\infty} \frac{A_k - k/\lambda}{\sigma\sqrt{k}} = N(0,1), \tag{4.1.6}$$

where $N(0,1)$ is a standard normal random variable with distribution $\Phi(\cdot)$. Similarly, for an α yet to be determined,[7]

$$\frac{A(t) - \lambda t}{\alpha\sqrt{t}} \to N(0,1). \tag{4.1.7}$$

Thus, $\mathsf{E}[A(t)] = \lambda t$ and $\mathsf{V}[A(t)] = \alpha^2 t$.

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

$$\Phi(y) \approx \mathsf{P}\left\{\frac{A_k - k/\lambda}{\sigma\sqrt{k}} \leq y\right\} \tag{4.1.8}$$

$$= \mathsf{P}\left\{\frac{A_k - k/\lambda}{\sigma\sqrt{k}} > -y\right\} \tag{4.1.9}$$

$$= \mathsf{P}\left\{A_k > \frac{k}{\lambda} - y\sigma\sqrt{k}\right\}. \tag{4.1.10}$$

We can use this relation between the distributions of $A(t)$ and A_k to see that $\mathsf{P}\{A_k > t_k\} = \mathsf{P}\{A(t_k) < k\}$ where we define for ease

$$t_k = \frac{k}{\lambda} - y\sigma\sqrt{k}. \tag{4.1.11}$$

With this we get,

$$\Phi(y) \approx \mathsf{P}\{A_k > t_k\} \tag{4.1.12}$$

$$= \mathsf{P}\{A(t_k) < k\} \tag{4.1.13}$$

$$= \mathsf{P}\left\{\frac{A(t_k) - \lambda t_k}{\alpha\sqrt{t_k}} < \frac{k - \lambda t_k}{\alpha\sqrt{t_k}}\right\}. \tag{4.1.14}$$

[5] *This can of course also be used to estimate C_s^2.*

[6] *It is based an argument in Cox [1962].*

[7] *This is a common trick: suppose that there exists a constant to take of the scaling, and then try an extra relation to 'ferret out' the scaling constant.*

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

$$\frac{k-\lambda t_k}{\alpha\sqrt{t_k}} \to y, \tag{4.1.15}$$

as $t_k \to \infty$. Using the above definition of t_k, the LHS of this equation can be written as

$$\frac{k-\lambda t_k}{\alpha\sqrt{t_k}} = \frac{\lambda\sigma\sqrt{k}}{\alpha\sqrt{k/\lambda+\sigma\sqrt{k}}}y. \tag{4.1.16}$$

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

$$\frac{\lambda\sigma\sqrt{k}}{\alpha\sqrt{k/\lambda+\sigma\sqrt{k}}}y \to y, \tag{4.1.17}$$

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

$$\alpha = \lambda^{3/2}\sigma. \tag{4.1.18}$$

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

$$\frac{\sigma_k^2}{\lambda_k} = \frac{\mathsf{V}[A(t)]}{\mathsf{E}[A(t)]} \approx \frac{\alpha^2 t}{\lambda t} = \frac{\alpha^2}{\lambda} = \frac{\lambda^3\sigma^2}{\lambda} = \lambda^2\sigma^2 = C_a^2, \tag{4.1.19}$$

where the last equation follows from the above definition of C_a^2.

Ex 4.1.1. In a manufacturing setting, the Poisson process is not always a suitable model for the arrival process of jobs at a production station. Can you provide an example to see why this is the case?

Ex 4.1.2. Consider a single-server queue at which every minute a customer arrives, precisely at the first second and $S \equiv 50$ s. What are ρ, $\mathsf{E}[L]$, C_a^2, and C_s^2?

This is an important example to memorize.

Ex 4.1.3. Consider the same single-server system as in [4.1.2], 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?

Ex 4.1.4. (Hall 5.19) When a bus reaches the end of its line, it undergoes a series of inspections. The entire inspection takes 5 minutes on average, with a standard deviation of 2 minutes. Buses arrive with inter-arrival times uniformly distributed on [3,9] minutes, hence, 10 buses arrive per hour on average. There is one mechanic available for the inspection. Use (4.1.1a) to estimate $\mathsf{E}[W]$.

Next, assuming that buses arrive as a Poisson process, estimate $\mathsf{E}[W]$.

Why is the queueing time smaller in the first setting?

Ex 4.1.5. Show for the *G/G/c/K* queue that $\beta = 1-\mathsf{E}[L_s]\mu/\lambda$, where $\mu = 1/\mathsf{E}[S]$ is the service rate, β the long-run fraction of customers lost, and $\mathsf{E}[L_s]$ the average number of busy/occupied servers.

Ex 4.1.6. 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 a 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[8]

$$\mathsf{E}[S] = p\,\mathsf{E}[S_1] + q\,\mathsf{E}[S_2] \tag{4.1.20}$$

$$\mathsf{V}[S] = p\,\mathsf{V}[S_1] + q\,\mathsf{V}[S_2] + pq(\mathsf{E}[S_1] - \mathsf{E}[S_2])^2. \tag{4.1.21}$$

[8] *Thus, even if* $\mathsf{V}[S_1] = \mathsf{V}[S_2] = 0$, *still* $\mathsf{V}[S] > 0$ *if* $\mathsf{E}[S_1] \neq \mathsf{E}[S_2]$. *Mixing different jobs types increases variability, hence queueing times.*

4.2 SETUPS AND BATCH PROCESSING

In some cases, machines have to be setup before they can start producing items. Consider, for instance, a machine that paints red and blue items.[0] When the machine requires a color change, it may be necessary to clean up the machine, which takes time. Another example is an oven that needs a temperature change when different item types require different production temperatures. Service operations form another setting with setup times: when servers (personnel) have to move from one part of a building to another, the time spent moving cannot be spent on serving customers.[1]

[0] *In realistic problems there can be hundreds of families.*

[1] *Often, the setup time depends on the sequence in which the families are produced, for example, a switch in color from white to black takes less cleaning time and cost than from black to white. The problem then becomes to determine first a production sequence that minimizes the sum of the setup times to produce the families, and then find suitable batch sizes to minimize the average waiting times. This problem becomes yet more realistic (and very hard) when jobs have due-dates too.*

In all such cases, the setups consume a significant amount of time; in fact, setup times of an hour or longer are not uncommon. Clearly, in such situations, it is necessary to produce in batches: a server processes a batch of jobs of one type or at one location, then changes from type or location, starts serving a batch of another type or at another location, and so on.

Here we focus on the effect of change-over, or setup, times on the average sojourn time of jobs. First we make a model and provide a list of elements required to compute the expected sojourn time of an item, then we illustrate how to use these elements in a concrete case.

ASSUME THERE ARE two job families, e.g., red and blue, each served by the same single server. Jobs arrive at rate λ_r and λ_b. The SCV of job inter-arrival times is given by C_a. Jobs of both type require the same average net processing time[2] of $\mathsf{E}[S_0]$, provided the server is already setup for the correct job color, and have variance $\mathsf{V}[S_0]$. Setup times are i.i.d. and have mean $\mathsf{E}[R]$ and variance $\mathsf{V}[R]$, and are assumed to be independent of job service times.

[2] *Net processing time*

A job's sojourn time is built up as follows. First, an arrival is assembled with other jobs of the same color to form a batch of constant size B. For simplicity, B is the same for both colors.[3] Once a batch is complete, the batch enters a queue (of batches). After some time the batch reaches the head of the queue, and is ready to move to the machine as soon as it becomes available. To serve a batch, the machine first performs a setup, and then processes each job individually until the batch is finished. Once a job's service is completed, it may, or may not[4], have to wait for the

[3] *In general, B should depend on the arrival rate of the job type. For instance, when red jobs arrive much faster than blue jobs, it takes much longer to fill a blue batch than a red batch when batch sizes are equal.*

[4] *Depending on the type of production.*

other jobs in the same batch before it can leave. Let us make a model of this.

WHEN A JOB OF TYPE i ARRIVES, the expected time for its batch to be formed is given by[5]

$$\mathsf{E}[W_i] = \frac{B-1}{2\lambda_i}, \quad i \in \{r, b\}. \tag{4.2.1}$$

When the batch is complete, it joins the queue, so we next compute the average time in queue.[6]

NOW THAT WE HAVE A batch of jobs, we need to estimate the average time a batch spends in queue. For this we can use Sakasegawa's formula; we only have to find expressions for each of its components.

The total arrival rate of jobs is $\lambda = \lambda_b + \lambda_r$. Thus, $\lambda_B = \lambda / B$ is the arrival rate of *batches*. It is easy to see that the expected service time of a batch is

$$\mathsf{E}[S_B] = \mathsf{E}[R] + B\,\mathsf{E}[S_0]. \tag{4.2.2}$$

Sometimes it useful to convert the effects of the setup time into an effective processing time[7] of an individual job:

$$\mathsf{E}[S] = \frac{\mathsf{E}[S_B]}{B} = \mathsf{E}[S_0] + \frac{\mathsf{E}[R]}{B}. \tag{4.2.3}$$

With the batch arrival rate and expected batch service time, the load becomes $\rho = \lambda_B\,\mathsf{E}[S_B]$

It is essential that B is sufficiently large to ensure that $\rho < 1$. With (4.2.2) this leads to the condition that B must be larger than some minimal batch size, i.e.,

$$B > B_m = \frac{\lambda\,\mathsf{E}[R]}{1 - \lambda\,\mathsf{E}[S_0]}. \tag{4.2.4}$$

Now that we have identified the service time of a batch and the load, we only need the squared coefficients of variation for the batch inter-arrival times $C^2_{a,B}$ and the batch service times[8] $C^2_{s,B}$ for Sakasegawa's formula. We find that[9]

$$C^2_{a,B} = \frac{C^2_a}{B}, \qquad C^2_{s,B} = \frac{\mathsf{V}[S_B]}{(\mathsf{E}[S_B])^2}, \tag{4.2.5}$$

where

$$\mathsf{V}[S_B] = B\,\mathsf{V}[S_0] + \mathsf{V}[R]. \tag{4.2.6}$$

By dividing by B we see that the variance of the effective processing time is

$$\mathsf{V}[S] = \mathsf{V}[S_0] + \frac{\mathsf{V}[R]}{B}. \tag{4.2.7}$$

IT IS LEFT TO FIND a rule to determine what happens to an item after it has been processed. If the job has to wait until all jobs in the batch are served, the expected time it spends at the server is $\mathsf{E}[R] + B\,\mathsf{E}[S_0]$. However, if the item can leave right after being served, the expected time at the server is [4.2.5]

$$\mathsf{E}[R] + \frac{B-1}{2}\mathsf{E}[S_0] + \mathsf{E}[S_0] = \mathsf{E}[R] + \frac{B+1}{2}\mathsf{E}[S_0]; \tag{4.2.8}$$

the first component is the average time a job has to wait before its service starts, the second is its service time. Our model is complete![10]

[5] *[4.2.2]*

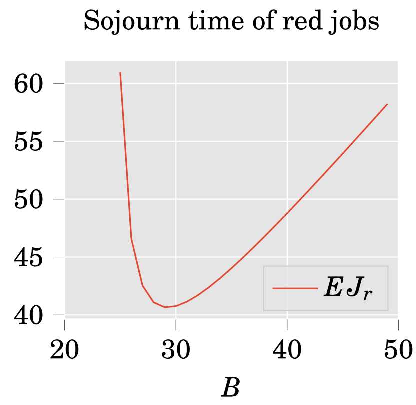
[6] *Note that we shift interpretation: batches (not individual items) form a queue and move as a whole to the server.*

[7] *Effective processing time*

[8] *Once again, for the* batches, *not the jobs!*

[9] *[4.2.3] [4.2.4]*

[10] *The model in [2.1.9] is more general as it allows the server to work at varying rates.*



WE CAN OBTAIN A NUMBER of important insights from the above model. Using [4.2.1] we plot the sojourn time $\mathsf{E}[J_r]$ of a red job for various values of B in the figure at the right.

First we see a sharp decline of $\mathsf{E}[J_r]$. The reason for this is that the load ρ decreases as a function of B. Since we know that $\mathsf{E}[W] \sim (1-\rho)^{-1}$, it is indeed essential to stay away from critically loading the server.

When B becomes quite large, we see that the sojourn time increases linearly. This follows right away from (4.2.1) and (4.2.8), because the time to (dis)assemble batches is linear in B.

Finally, observe that the graph is not symmetric around the minimum. It is much worse to take B too small than too large.

Before dealing with technical derivations, let us first see how to apply the model.

Ex 4.2.1. Jobs arrive at $\lambda = 3$ per hour at a machine with $C_a^2 = 1$; service times are exponential with an average of 15 minutes. Assume $\lambda_r = 0.5$ per hour, hence $\lambda_b = 2.5$ per hour. Between any two batches, the machine requires a cleanup of 2 hours, with a standard deviation of 1 hour. Fist find the smallest batch size that can be allowed, then compute the average time a red job spends in the system in case $B = 30$ jobs.

Ex 4.2.2. Show that the average time a job has to wait to fill the batch (to which this job belongs) is given by (4.2.1).

Ex 4.2.3. Explain that the SCV of the batch inter-arrival times is given by (4.2.5).

Ex 4.2.4. Show that the SCV $C_{s,B}^2$ of the service times of the batches takes the form as in (4.2.5).

Ex 4.2.5. Show that, when items can leave right after being served, the time at the server is given by (4.2.8)

4.3 SERVER ADJUSTMENTS

In Section 4.2 we studied the effect of setup times between batches with constant size B. However, the server can be interrupted for other reasons than setups. For instance, parts in a machine, such as knives, may need to be replaced or adjusted. Such small tasks can be carried out in between jobs, but it can be hard to predict when it is required.[0] Another example is a GP who to make a unexpected phone calls between two seeing two patients. In all such cases, the number of jobs[1] served between any two adjustments[2] is no longer constant. But we know from Sakasegawa's formula that randomness in service times affects queueing times, and we also know from (4.2.3) that the batch size affects the service time. Hence, unplanned adjustments must have an effect on sojourn times.

In this section we develop a simple model to understand the impact of such outages on job sojourn times; we use the same notation as in Section 4.2 and follow the same line of reasoning. With this model, we can analyze a number of trade-offs, such doing fewer, but longer adjustments, or planning adjustments instead just waiting until it becomes necessary at an unexpected moment.[3] With the model we can make graphs of the

[0] For instance, the breaking of a chisel is a random event.

[1] You can compare this to a batch of jobs.

[2] And this to a setup.

[3] [4.3.2]

sojourn time as a function of adjustment rate, for instance, so that we can optimize for the adjustment rate.

It is important to realize that setups and adjustments are both types of non-preemptive outages[4], i.e., they occur *between* jobs, not *during* job service times.

We assume that run lengths $\{B_i\}$, i.e., the number of jobs served between two adjustments, forms a sequence of geometric i.i.d. random variables such that $\mathsf{P}\{B=k\}=(1-p)^{k-1}p$, where p is the 'success' probability.[5] As a consequence, $\mathsf{E}[B]=1/p$ and variance $\mathsf{V}[B]=(1-p)/p^2$.[6] Suppose also that the adjustment (repair) times are i.i.d. with mean $\mathsf{E}[R]$ and variance $\mathsf{V}[R]$.

To compute $\mathsf{E}[W]$ we only have to find an expression how the adjustments affect the mean and SCV C_s^2 of the processing times. All other terms in the $G/G/1$ waiting time formula remain the same, as the adjustments do not affect the job arrival process, It is simple[7] to see that the average effective processing time is

$$\mathsf{E}[S]=\mathsf{E}[S_0]+p\,\mathsf{E}[R]=\mathsf{E}[S_0]+\frac{\mathsf{E}[R]}{\mathsf{E}[B]}. \tag{4.3.1}$$

Consequently, the effective[8] load becomes $\lambda\,\mathsf{E}[S]$, which is equal to ρ since there one server. With this and an expression[9] for $\mathsf{E}\left[S^2\right]$, we find[10] after some polishing

$$\begin{aligned}\mathsf{V}[S]&=\mathsf{V}[S_0]+p\,\mathsf{V}[R]+p(1-p)(\mathsf{E}[R])^2\\&=\mathsf{V}[S_0]+\frac{\mathsf{V}[R]+(\mathsf{E}[R])^2C_B^2}{\mathsf{E}[B]},\end{aligned} \tag{4.3.2}$$

where C_B^2 is the SCV of the runlength B. We can now fill in Sakasegewa's formula.

LET US COMPARE the results of this section to those of Section 4.3. In both cases the expected service time is the same: an amount $\mathsf{E}[R]/\mathsf{E}[B]$ gets added to the net processing time $\mathsf{E}[S_0]$. However, the impact on the variance is different. By comparing (4.2.7) to (4.3.2) we see that the former is smaller. This supports our intuition: Unexpected interruptions must have a larger effect on variability than expected interruptions.

Ex 4.3.1. Jobs arrive as a Poisson process with rate $\lambda=9$ per working day. The machine works two 8 hour shifts a day. Work not processed on a day is carried over to the next day. Job service times are 1.5 hours, on average, with standard deviation of 0.5 hours. Outages occur on average between 30 jobs. The average duration of an outage is 5 hours and has a standard deviation of 2 hours. Compute $\mathsf{E}[J]$.

We first show how to apply the model before we deal with the derivations.

Ex 4.3.2. In the setting of [4.3.1] we can perhaps choose to do an adjustment after every 20 jobs. For simplicity, assume that then adjustments never occur at random. Also assume that an adjustment takes less time, for instance 4.5 hour and are constant. What is $\mathsf{E}[J]$ now?

In maintenance this is a common problem. Should we wait until a failure occurs, or should we plan the repairs?

Because planned adjustment can be prepared.

Ex 4.3.3. Derive (4.3.1).

[4] *Non-preemptive outages*

[5] *Recall from (2.4.4) that the run-length B is memoryless under these assumptions.*

[6] *To obtain p in practice, we measure the number of jobs served between a number of adjustments, and take the mean.*

[7] *[4.3.3]*

[8] *Hence, including down-times!*

[9] *[4.3.4]*

[10] *[4.3.5]*

Ex 4.3.4. Show that

$$\mathsf{E}\left[S^2\right] = \mathsf{E}\left[S_0^2\right] + 2p\,\mathsf{E}[S_0]\,\mathsf{E}[R] + p\,\mathsf{E}\left[R^2\right]. \tag{4.3.3}$$

Ex 4.3.5. Derive (4.3.2).

4.4 SERVER FAILURES

In Sections 4.2 and 4.3 we assumed that servers are never interrupted while serving a job. However, this assumption is not always satisfied, for instance, a machine may fail in the midst of processing of a job. In this section, we develop a model[0] to compute the influence on the mean waiting time of such *preemptive outages*, again based on Sakasegawa's formula for the $G/G/1$ queue.

As in the previous sections, we derive expressions for the expectation and variance of the effective processing time S.

Supposing that N interruptions occur during the net service time S_0 of a job, and the repair times $\{R_i\}$ form an i.i.d. sequence whose common distribution is that of R, the effective service time becomes[1]

$$S = S_0 + S_N = S_0 + \sum_{i=1}^{N} R_i. \tag{4.4.1}$$

In general, N is a random number, so we need to adjust for this in the computation of the mean and variance of S.

IT IS COMMON to assume that the time between two interruptions is memoryless with mean $1/\lambda_f$, where λ_f is the *failure rate*. Then, if the net service time S_0 is a constant, the expected number of failures $\mathsf{E}[N]$ is $\lambda_f S_0$. More generally, if S_0 is also a random variable, we find[2] that $\mathsf{E}[N] = \lambda_f\,\mathsf{E}[S_0]$.

Define the *availability* as

$$A = \frac{m_f}{m_f + \mathsf{E}[R]} \tag{4.4.2}$$

where m_f is the mean time to fail. Since, $m_f = 1/\lambda_f$, it is easy to show[3] that $A = (1+\lambda_f\,\mathsf{E}[R])^{-1}$, from which we obtain [4] $\mathsf{E}[S] = \mathsf{E}[S_0]/A$. With this, the utilization becomes

$$\rho = \lambda\,\mathsf{E}[S] = \lambda\frac{\mathsf{E}[S_0]}{A}. \tag{4.4.3}$$

Since $A \in (0,1)$, the server load increases due to failures.

BY ASSUMING THAT repair times are exponentially distributed, we find[5] that

$$C_s^2 = C_0^2 + 2A(1-A)\frac{\mathsf{E}[R]}{\mathsf{E}[S_0]}, \tag{4.4.4}$$

where C_0^2 is the SCV of S_0.

Ex 4.4.1. Suppose we have a machine with memoryless failure behavior, with a mean-time-to-fail of 3 hours. Regular service times are deterministic with an average of 10 minutes, jobs arrive as a Poisson process with rate of 4 per hour. Repair times are exponential with a mean duration of 30 minutes. What is the average sojourn time?

[0] *There is not much point in combining the models of Section 4.2–Section 4.4 into one large model. For instance, when a machine can be setup, while a part is being repaired, there may not be time lost on the setup. All depends on the specific strategies to combine outages.*

[1] *If we interpret $\{R_i\}$ as a set of claims, then $\sum_{i=1}^N R_i$ is the total claim size of N claims. Therefore, insurance companies are also interested in the characterization of this object. Likewise, In the setting of inventory systems, it is the total demand of N customers.*

[2] *[4.4.5]*

[3] *[4.4.2]*

[4] *[4.4.6]*

[5] *[4.4.7]–[4.4.15]*

Ex 4.4.2. Derive (4.4.2) for our model of interruptions.

Ex 4.4.3. Suppose that the number of failures is equal to the number n, show that $\mathsf{E}\left[\sum_{i=1}^n R_i\right] = n\,\mathsf{E}[R]$.

Ex 4.4.4. Show that $\mathsf{E}\left[\sum_{i=1}^N R_i\right] = \mathsf{E}[R]\,\mathsf{E}[N]$ when N is random variable with finite expectation.

This result is known as Wald's equation.

Ex 4.4.5. Show that $\mathsf{E}[N] = \lambda_f\,\mathsf{E}[S_0]$ and $\mathsf{E}\left[N^2\right] = \lambda_f^2\,\mathsf{E}\left[S_0^2\right] + \lambda_f\,\mathsf{E}[S_0]$ if we assume for the sake of simplicity that S_0 has the density g,

[4.4.5]–[4.4.10] are of fundamental importance in insurance, inventory and queueing theory. They also provide relations between the topics discussed in Section 2.2, Section 6.6, and Section 7.2.

Ex 4.4.6. Show that $\mathsf{E}[S] = \mathsf{E}[S_0] + \lambda_r\,\mathsf{E}[S_0]\,\mathsf{E}[R]$, and conclude that $\mathsf{E}[S] = \mathsf{E}[S_0]/A$.

Ex 4.4.7. The derivation of C_s^2 is a bit more involved. To understand why, explain first that $\mathsf{V}[S] \neq \mathsf{V}[S_0] + \mathsf{V}\left[\sum_{i=0}^N R_i\right]$.

Ex 4.4.8. Show that

$$\mathsf{E}\left[S^2\right] = \mathsf{E}\left[S_0^2\right] + 2\,\mathsf{E}\left[S_0 \sum_{i=1}^N R_i\right] + \mathsf{E}\left[\sum_{i=1}^N R_i^2\right] + \mathsf{E}\left[\sum_{i=1}^N \sum_{j\neq i} R_i R_j\right]. \tag{4.4.5}$$

Ex 4.4.9. Show that $\mathsf{E}\left[S_0 \sum_{i=1}^N R_i\right] = \lambda_f\,\mathsf{E}[R]\,\mathsf{E}\left[S_0^2\right]$.

Ex 4.4.10. Show that $\mathsf{E}\left[\sum_{i=1}^N R_i^2\right] = \lambda_f\,\mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right]$.

Ex 4.4.11. Show that $\mathsf{E}\left[\sum_{i=1}^N \sum_{j\neq i} R_i R_j\right] = \lambda_f^2\,\mathsf{E}\left[S_0^2\right](\mathsf{E}[R])^2$.

Ex 4.4.12. Combine the above to see that

$$\mathsf{E}\left[S^2\right] = \frac{\mathsf{E}\left[S_0^2\right]}{A^2} + \lambda_f\,\mathsf{E}\left[R^2\right]\mathsf{E}[S_0]. \tag{4.4.6}$$

The above exercises suffice for the computations. With $\mathsf{E}\left[S^2\right]$ and $\mathsf{E}[S]$ we can compute the SCV via $(\mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2)/(\mathsf{E}[S])^2$. The rest is polishing.

Ex 4.4.13. Show that

$$\mathsf{V}[S] = \frac{\mathsf{V}[S_0]}{A^2} + \lambda_f\,\mathsf{E}\left[R^2\right]\mathsf{E}[S_0]. \tag{4.4.7}$$

Ex 4.4.14. Show that

$$C_s^2 = \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} = C_0^2 + \frac{\lambda_f\,\mathsf{E}\left[R^2\right]A^2}{\mathsf{E}[S_0]}, \tag{4.4.8}$$

Ex 4.4.15. With the above assumption that R is exponentially distributed, show that

$$C_s^2 = C_0^2 + 2A(1-A)\frac{\mathsf{E}[R]}{\mathsf{E}[S_0]}. \tag{4.4.9}$$

4.5 $G/G/1$ QUEUES IN TANDEM

Consider two $G/G/1$ stations in tandem.[0] Suppose we have the financial means to reduce the variability of the processing times at one of the stations, but not at both. Then we like to improve the one that has the most impact on the total waiting time in the line.

[0] 'In tandem' means 'in line', one station after the other.

For the waiting time of the first machine, we can use Sakasegawa's formula, but to apply this to the second machine, we need $C_{a,2}^2$, i.e., the

SCV of the inter-arrival times at the second station. Now, noting that the output of the first machine forms the input of the second machine, it is clear that $C^2_{a,2} = C^2_{d,1}$, where $C^2_{d,1}$ is the SCV of the *inter-departure* times of *first* station.

In this section, we present a formula[1] to approximate $C^2_{d,1}$, and with this, we can model the propagation of variability through a tandem network of $G/G/c$ stations.

[1] *The literature provides algorithms to deal with networks of $G/G/1$ queues in which the output of several stations form the input of another station and rework is allowed. In Section 7.3 we analyze such networks, but only for $M/M/c$ stations.*

Let us consider the inter-departure times of a $G/G/1$ queue. Suppose that the utilization ρ is very high. Then the server will seldom be idle, so that most of the inter-departure times are the same as service times. If, however, the utilization is low, the server will be idle most of the time, and the inter-departure times must be approximately equal to the inter-arrival times. We obtain an approximation for the SCV C^2_d of the inter-departure times by interpolating between these two extremes[2]:

$$C^2_d \approx (1-\rho^2)C^2_a + \rho^2 C^2_s. \tag{4.5.1}$$

[2] *For the $G/G/c$ there is the generalization $C^2_d \approx 1+(1-\rho^2)(C^2_a-1)+\frac{\rho^2}{\sqrt{c}}(C^2_s-1)$. It is simple to see that this reduces to (4.5.1).*

Combining Sakasegawa's formula with this expression provides us with a very useful insight for a line. If we reduce $C^2_{s,1}$, i.e., the SCV of service times at the first station, $\mathsf{E}[W_1]$ and $C^2_{d,1}$ become smaller. Since $C^2_{a,2} = C^2_{d,1}$, $\mathsf{E}[W_2]$ becomes lower too, but also $C^2_{d,2}$, and so on. In other words, the entire chain benefits from an improvement in service variability at the first station.[3]

[3] *Try to improve at the start.*

Ex 4.5.1. Consider two $G/G/1$ stations in tandem. Suppose $\lambda = 2$ per hour, $C^2_{a,1} = 2$, $C^2_{s,1} = C^2_{s,2} = 0.5$, and $\mathsf{E}[S_1] = 20$ minutes and $\mathsf{E}[S_2] = 25$ minutes. What is $\mathsf{E}[J] = \mathsf{E}[J_1] + \mathsf{E}[J_2]$?

5

FUNDAMENTAL TOOLS

To develop mathematical models of queueing systems we need a few concepts that are fundamentally important and have a general interest beyond queueing. All these concepts rely on *sample-path constructions* of queueing, or more general stochastic, systems. We will see that sample paths, which are in fact realizations of simulations of queueing systems, form an elegant and unifying principle.

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

5.1 RENEWAL REWARD THEOREM

We introduce the renewal reward theorem and provide graphical motivation for its validity.[0] This result proves to be extremely useful: we will apply it regularly in the sequel of the book and here we use it to provide another definition of the utilization ρ.

[0] *El-Taha and Stidham Jr. [1998] give a (simple) proof based on similar arguments as used in Section 3.3.*

In Fig. 9, consider a strictly increasing set of epochs $\{T_k, k = 0, 1, \ldots\}$ and let $N = \{N(t), t \geq 0\}$ be the associated counting process such that $N(t) = \max\{k : T_k \leq t\}$. Let $\{Y(t), t \geq 0\}$ be a non-decreasing right-continuous (deterministic) process, and define $X_k = Y(T_k) - Y(T_{k-1})$, i.e,. as the increment[1] in Y between the epochs T_{k-1} and T_k.

[1] *Here X_k is not an inter-arrival time between jobs.*

The renewal reward theorem[2] states that when $\lim_{t\to\infty} N(t)/t = \lambda$ with $0 < \lambda < \infty$, $Y(t)/t$ has a limit iff $n^{-1}\sum_{k=1}^{n} X_k$ has a limit, and then $Y = \lambda X$.[3]

[2] *Renewal reward theorem*

[3] *In essence the renewal reward theorem is very simple: it states that when* customers *arrive at rate λ and each* customer *pays an average amount X, the* system *earns money at rate $Y = \lambda X$.*

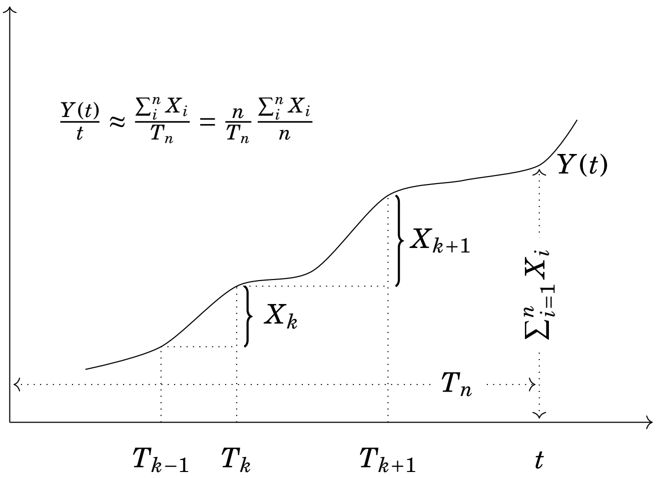


Figure 9: A graphical 'proof' of $Y = \lambda X$.

WITH THE RENEWAL REWARD THEOREM we can relate limiting fraction of time the server is busy to the utilization $\rho = \lambda \mathsf{E}[S]$. In fact, for the $G/G/1$ queue,

$$\lim_{t\to\infty} \frac{1}{t}\int_0^t \mathbb{1}_{L(s)>0}\,\mathrm{d}s = \lambda \mathsf{E}[S] = \rho. \tag{5.1.1}$$

We can derive this relation also by considering the fact that

$$\sum_{k=1}^{A(t)} S_k \geq \int_0^t \mathbb{1}_{L(s)>0}\,\mathrm{d}s \geq \sum_{k=1}^{D(t)} S_k. \tag{5.1.2}$$

By taking appropriate limits[4]

$$\lambda \mathsf{E}[S] \geq \rho \geq \delta \mathsf{E}[S]. \tag{5.1.3}$$

Rate-stability, i.e., $\delta = \lambda$, then completes the argument.

[4] [5.1.2]

Ex 5.1.1. Use the renewal reward theorem to prove (5.1.1).

Ex 5.1.2. Derive (5.1.3).

5.2 LEVEL CROSSING AND BALANCE EQUATIONS

Consider a sample path[0] of $\{L\}$, that is, a sample path of $\{L\}$. We say that the sample path up-crosses[1] level n when L changes from n to $n+1$ due to an arrival, and it down-crosses[2] level n when L changes from $n+1$ to n due to a departure. Clearly, the number of up-crossings and down-crossings cannot differ by more than 1 at any time, because it is only possible to up-cross level n after a down-crossing (or the other way around). This simple idea will prove a key stepping stone in the analysis of queueing systems.

[0] The graph of the number of jobs in the system obtained by simulation.

[1] Up-crosses

[2] Down-crosses

To establish the section's main result (5.2.9), we need a few definitions that are quite subtle, but we will provide intuitive and natural interpretations.[3] After this, we will generalize the principle of level-crossing to *balance equations* which allow us to deal with more general types of transitions.[4]

[3] They have practical significance too.

[4] See Fig. 10 for a graphical summary.

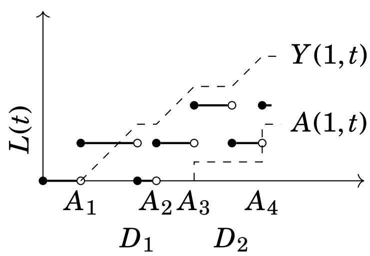
Let us say that the system is in *state* n at time t when $L(t) = n$. Then define[5]

$$A(n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t}\, \mathbb{1}_{L(A_k-)=n} \tag{5.2.1a}$$

as the number of arrivals up to time t that saw the system in state n.[6] Next, let

[5] Note the A_k-; since $L(t)$ is right-continuous, we concentrate on what an arrival *sees.*

[6] [5.2.1],[5.2.2]



We subtracted 1/2 from the graph of $A(1,t)$, for otherwise this overlaps with the graph of Y.

$$Y(n,t) = \int_0^t \mathbb{1}_{L(s)=n}\,\mathrm{d}s \tag{5.2.1b}$$

be the total time the system spends in state n during $[0,t]$, and

$$p(n,t) = \frac{1}{t}\int_0^t \mathbb{1}_{L(s)=n}\,\mathrm{d}s = \frac{Y(n,t)}{t}, \tag{5.2.1c}$$

be the fraction of time in state n during $[0,t]$.

With the above definitions, we can consider the limits,[7]

$$\lambda(n) = \lim_{t\to\infty} \frac{A(n,t)}{Y(n,t)}, \qquad p(n) = \lim_{t\to\infty} p(n,t), \tag{5.2.2}$$

[7] Again assuming they exist.

as the arrival rate while the system is in state n, and the long-run fraction of time spent in state n. To see the rationale behind this definition of $\lambda(n)$, compare $A(t)/t$, which is the total number of arrivals during $[0,t]$ divided by t. Likewise, $A(n,t)/Y(n,t)$ is the number of arrivals that see n jobs divided by the time the system contains n jobs.

Similar to the definition for $A(n,t)$, let[8]

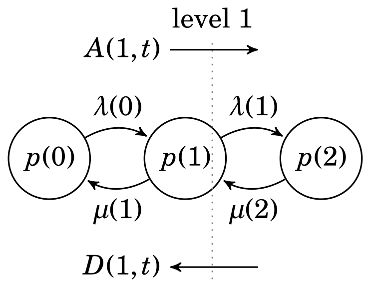
$$D(n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \le t} \mathbb{1}_{L(D_k)=n} \tag{5.2.3}$$

denote the number of departures up to time t that *leave n customers behind*. Then, define[9]

$$\mu(n+1) = \lim_{t\to\infty} \frac{D(n,t)}{Y(n+1,t)}, \tag{5.2.4}$$

as *the departure rate from state* $n+1$.

[8] *But now we take D_k, not D_k-.*



[9] *It is easy to get confused here: to leave n jobs behind, the system must contain $n+1$ jobs just prior to the departure.*

THE ABOVE DEFINITIONS may seem a bit abstract, but they have simple and practical interpretations.

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. Incoming parcels are deposited on a conveyor belt, and from there, they are carried to outlets, so-called chutes, from where the parcels are sent to a specific region of the Netherlands. Employees pick up the parcels from the chutes and put the parcels in containers. Sometimes parcels arrive a bit faster than the capacity of the employees and then a queue of parcels builds up in the chute. 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 99% of the parcels within one day.

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 sent to the overflow container. Clearly then, $A(20,t)/A(t)$ is the fraction of rejected parcels up to time t. For this reason, sorting centers continuously track $A(20,\cdot)$ to adapt server capacity and control the fraction of rejected parcels.

For a second example, suppose it costs w to have a job in queue for a unit of time. The total cost up to time t is then $w\sum_{n=0}^{\infty} nY(n,t)$ and the average cost is $w\sum_{n=0}^{\infty} np(n,t)$.

CONTINUING WITH THE theoretical discussion, observe that customers arrive and depart as single units. Thus, if $\{T_k\}$ is the ordered set of arrival and departure epochs of the customers, then $L(T_k) = L(T_k-) \pm 1$. But this implies that[10]

$$|A(n,t) - D(n,t)| \le 1. \tag{5.2.5}$$

[10] *Think about this. This is simple, but has profound consequences.*

From this observation it follows immediately that

$$\lim_{t\to\infty} \frac{A(n,t)}{t} = \lim_{t\to\infty} \frac{D(n,t)}{t}. \tag{5.2.6}$$

With this equation we obtain two nice identities. The first we develop here, the other in Section 5.3.

Clearly,[11] $\lim_{t\to\infty} A(n,t)/t$ is the rate of arrivals that see the system in state n. With (5.2.1) we see that

$$\lim_{t\to\infty} \frac{A(n,t)}{t} = \lim_{t\to\infty} \frac{A(n,t)}{Y(n,t)} \frac{Y(n,t)}{t} = \lambda(n)p(n). \tag{5.2.7}$$

Similarly, the rate of jobs that leave n jobs behind is

$$\lim_{t\to\infty} \frac{D(n,t)}{t} = \lim_{t\to\infty} \frac{D(n,t)}{Y(n+1,t)} \frac{Y(n+1,t)}{t} = \mu(n+1)p(n+1). \tag{5.2.8}$$

Combining this with (5.2.6) we arrive at the level-crossing equations[12,13]

$$\lambda(n)p(n) = \mu(n+1)p(n+1). \tag{5.2.9}$$

SUPPOSE WE CAN SPECIFY[14] the arrival and service rates $\lambda(n)$ and $\mu(n)$, then we can easily compute the long-run fraction of time $p(n)$ that the system contains n jobs. To see this, rewrite (5.2.9) as

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)} p(n). \tag{5.2.10}$$

A straightaway recursion then leads to

$$p(n+1) = \frac{\lambda(n)\lambda(n-1)\cdots\lambda(0)}{\mu(n+1)\mu(n)\cdots\mu(1)} p(0). \tag{5.2.11}$$

Thus, $p(n)$, $n \geq 1$, is just $p(0)$ times a constant, which is based on arrival and service rates.

To determine $p(0)$ we can use the fact that the numbers $p(n)$ represent probabilities. Hence, from the normalizing condition $\sum_{n=0}^{\infty} p(n) = 1$, we get $p(0) = G^{-1}$ with G being the *normalization constant*

$$G = 1 + \sum_{n=0}^{\infty} \frac{\lambda(n)\lambda(n-1)\cdots\lambda(0)}{\mu(n+1)\mu(n)\cdots\mu(1)}. \tag{5.2.12}$$

Finally, once we have $p(n)$, it is easy to compute the time-average[15] number of jobs in the system and the long-run fraction of time the system contains at least n:

$$\mathsf{E}[L] = \sum_{n=0}^{\infty} n p(n), \qquad \mathsf{P}\{L \geq n\} = \sum_{i=n}^{\infty} p(i). \tag{5.2.13}$$

IT IS IMPORTANT TO REALIZE that the level-crossing argument cannot always be used, as it is not always possible to split the state space into two disjoint parts by 'drawing a line' between two states. For a more general approach, we focus on a single state and count how often this state is entered and left. Specifically, define $I(n,t) = A(n-1,t) + D(n,t)$ as the number of times the queueing process enters state n either due to an arrival from state $n-1$ or due to a departure leaving n jobs behind. Similarly, $O(n,t) = A(n,t) + D(n-1,t)$ counts how often state n is left either by an arrival (to state $n+1$) or a departure (to state $n-1$).

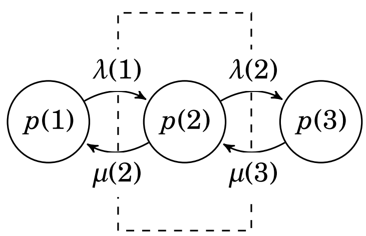
[11] *Supposing the limits exist.*

[12] *Level-crossing equations*

[13] *This result is of fundamental importance in queueing (and inventory) theory.*

[14] *In Section 6.1 and onward we will model many queueing situations by making suitable choices for $\lambda(n)$ and $\mu(n)$.*

[15] *It is important to realize that this is not necessarily the same as what jobs see upon arrival.*



Just like (5.2.5), it is evident that $|I(n,t) - O(n,t)| \le 1$, and this implies[16]

$$\lambda(n-1)p(n-1) + \mu(n+1)p(n+1) = (\lambda(n) + \mu(n))p(n). \qquad (5.2.14)$$

These equations hold for any $n \ge 1$ and are known as the balance equations[17]. We will use these equations to analyze queueing networks.

[16] [5.2.10]

[17] Balance equations

Ex 5.2.1. Show that $A(t) = \sum_{n=0}^{\infty} A(n,t)$.

Ex 5.2.2. If $\lambda > \delta$ can it happen that $\lim_{t\to\infty} A(n,t)/t > 0$ for some (finite) n?

Ex 5.2.3. Consider the following (silly) queueing process. At times $0, 2, 4, \ldots$ customers arrive, each customer requires 1 unit of service, and there is one server. Find an expression for $A(n,t)$, when $L(0) = 0$.

What acronym would describe this queueing situation?

Ex 5.2.4. Find an expression for $Y(n,t)$.

Continuation of [5.2.3]

Ex 5.2.5. Compute $p(n)$ and $\lambda(n)$.

Continuation of [5.2.4]

Ex 5.2.6. Compute $D(n,t)$ and $\mu(n+1)$ for $n \ge 0$.

Continuation of [5.2.5]

Ex 5.2.7. Compute $\lambda(n)p(n)$ for $n \ge 0$, and check $\lambda(n)p(n) = \mu(n+1)p(n+1)$.

Continuation of [5.2.6]

Ex 5.2.8. Derive $\mathsf{E}[L] = \sum_{n=0}^{\infty} np(n)$ from (3.4.4).

Ex 5.2.9. 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). At most 3 jobs fit in the system. Single jobs arrive as a Poisson process with λ. Due to blocking, we take $\lambda(n) = \lambda$ for $n < 3$ and $\lambda(n) = 0$ for $n \ge 3$. The batch service times are exponentially distributed with mean $1/\mu$, so that by the memoryless property, $\mu(n) = \mu$.

Make a graph of the state-space and show, with arrows, the transitions that can occur and use level-crossing arguments to express the steady-state probabilities $p(n), n = 0, \ldots, 3$ in terms of λ and μ.

When level-crossing arguments can be applied, the analysis often becomes quite easy.

What is the acronym for this system?

Ex 5.2.10. Show (5.2.14) from $|I(n,t) - O(n,t)| \le 1$.

5.3 POISSON ARRIVALS SEE TIME AVERAGES

Suppose jobs arrive exactly at the start of an hour and require 59 minutes of service. If we sample the server occupation at job arrival times, the server is always free, while if we track the server over time, it is nearly always busy. Thus, what jobs see upon arrival is *in general not equal* to what the server perceives. However, when jobs arrive as a Poisson process, both sampling methods produce the same number, a result known as the Poisson arrivals see time averages[0] (PASTA[1]) property. Here we will discuss this property in more detail.

[0] Poisson arrivals see time averages

[1] PASTA

Recall from Section 5.2 that

$$\frac{A(n,t)}{t} = \frac{A(n,t)}{Y(n,t)} \frac{Y(n,t)}{t} \to \lambda(n)p(n), \quad \text{as } t \to \infty. \qquad (5.3.1a)$$

Rather than multiplying and dividing the LHS by $Y(n,t)$, we can also multiply and divide by $A(t)$ to get another interesting result:

$$\frac{A(n,t)}{t} = \frac{A(t)}{t}\frac{A(n,t)}{A(t)}. \tag{5.3.1b}$$

It is clear that $A(t)/t \to \lambda$, so let us interpret $A(n,t)/A(t)$. For this, observe first

$$\frac{A(n,t)}{A(t)} = \frac{1}{A(t)}\sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t, L(A_k-)=n} = \frac{1}{A(t)}\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=n}. \tag{5.3.2}$$

Then, since $A(t) \to \infty$ as $t \to \infty$, it follows[2] that

[2] [5.3.4]

$$\lim_{t\to\infty} \frac{1}{A(t)}\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=n} = \lim_{m\to\infty} \frac{1}{m}\sum_{k=1}^{m} \mathbb{1}_{L(A_k-)=n} =: \pi(n), \tag{5.3.3}$$

where $\pi(n)$ is the long-run fraction of arrivals that observe n customers in the system.

But, in (5.3.1) the LHSs are equal. Hence,

$$\lambda\pi(n) = \lambda(n)p(n), \tag{5.3.4}$$

which implies the main result:

$$\lambda(n) = \lambda \iff \pi(n) = p(n). \tag{5.3.5}$$

In words, if the arrival rate does not depend on the state of the system, i.e., $\lambda(n) = \lambda$, the sample probabilities $\{\pi(n)\}$ are equal to the time-average probabilities $\{p(n)\}$.[3]

[3] *Thus, what arrivals see agrees with what the server sees.*

As the above example showed, this property is not satisfied in general. However, when the arrival process is Poisson we have[4] that $\lambda(n) = \lambda$, and its implication $\pi(n) = p(n)$ is known as the *PASTA* property.

[4] *A rigorous proof of this is hard, see e.g., El-Taha and Stidham Jr. [1998]*

WITH SIMILAR REASONING, we can also establish a relation between $\pi(n)$ and $\delta(n)$, i.e., statistics as obtained by the departures. Define, analogous to $\pi(n)$,

$$\delta(n) = \lim_{t\to\infty} \frac{D(n,t)}{D(t)} \tag{5.3.6}$$

as the long-run fraction of jobs that leave n jobs *behind*. From (5.2.6),

$$\frac{A(t)}{t}\frac{A(n,t)}{A(t)} = \frac{A(n,t)}{t} \approx \frac{D(n,t)}{t} = \frac{D(t)}{t}\frac{D(n,t)}{D(t)}. \tag{5.3.7}$$

Taking limits at the left and right, and using (3.3.2), we obtain for the $G/G/c$ queue[5]

[5] *Because customers arrive and leave as single units in a $G/G/c$ queue.*

$$\lambda\pi(n) = \delta\delta(n). \tag{5.3.8}$$

Consequently, for the rate-stable $G/G/c$ queue the statistics obtained by arrivals is the same as statistics obtained by departures, i.e.,

$$\lambda = \delta \iff \pi(n) = \delta(n). \tag{5.3.9}$$

Ex 5.3.1. Show that $\pi(0) = 1$ and $\pi(n) = 0$, for $n > 0$.

Continuation of [5.2.3]

Ex 5.3.2. Check that (5.3.4) holds.

Continuation of [5.3.1]

Ex 5.3.3. When $\lambda \neq \delta$, is $\pi(n) \geq \delta(n)$?

Ex 5.3.4. Use the renewal-reward theorem to prove that $A(n,t)/t \to \lambda\pi(n)$.

5.4 LITTLE'S LAW

There is an important relation between the average sojourn time of a job and the long-run time-average number of jobs contained in the system. This relation is called *Little's law*, and is one of the most useful results in queueing theory.[0] The aim of this section is to prove this law under some simple conditions.

[0] In fact, to analyze throughput in any input-output system.

We start by defining a few intuitively useful concepts. From (2.3.10) we see that

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \frac{1}{t}\int_0^t (A(s)-D(s))\,\mathrm{d}s \tag{5.4.1}$$

is the time-average of the number of jobs in the system during $[0,t]$. Next,[1] the sojourn time of the kth job is the time between the moment the job arrives and departs, that is,

[1] Check Fig. 4 to see how J_k and $L(t)$ relate.

$$J_k = \int_0^\infty \mathbb{1}_{A_k \le s < D_k}\,\mathrm{d}s. \tag{5.4.2}$$

Consider a departure time T at which the system is empty so that $A(T)=D(T)$. Then, for $k \le A(T)$,

$$J_k = \int_0^T \mathbb{1}_{A_k \le s < D_k}\,\mathrm{d}s, \tag{5.4.3}$$

and for $s \le T$,

$$L(s) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le s < D_k} = \sum_{k=1}^{A(T)} \mathbb{1}_{A_k \le s < D_k}. \tag{5.4.4}$$

Now realize that at time T the areas $\int_0^T L(s)\,\mathrm{d}s$ and $\sum_{k=1}^{A(T)} J_k$ are the same,[2] that is,

[2] [5.4.5]

$$\int_0^T L(s)\,\mathrm{d}s = \sum_{k=1}^{A(T)} J_k. \tag{5.4.5}$$

Assuming that there exists an infinite number of times $\{T_i\}$, $T_i \to \infty$, such that the system is empty, i.e., $A(T_i)=D(T_i)$, it is then evident that

$$\frac{1}{T_i}\int_0^{T_i} L(s)\,\mathrm{d}s = \frac{A(T_i)}{T_i}\frac{1}{A(T_i)}\sum_{k=1}^{A(T_i)} J_k. \tag{5.4.6}$$

Taking limits[3] then gives Little's law[4]:

[3] Provided they exist.

[4] Little's law

$$\mathsf{E}[L] = \lambda\,\mathsf{E}[J]. \tag{5.4.7}$$

Note that Little's law need not hold at all moments in time,[5] it is a statement about *averages*.

[5] [5.4.3] and [5.4.4]

WITH THE PASTA PROPERTY and Little's law it becomes quite easy to derive simple expressions for EJ and $\mathsf{E}[W]$ for the $M/M/1$ queue.[6] With PASTA, it follows for the $M/M/1$ queue that

[6] For the $M/G/1$ queue, things are bit subtler, see [5.4.6] and Section 6.4.

$$\mathsf{E}[J] = \mathsf{E}[L]\,\mathsf{E}[S]. \tag{5.4.8}$$

Combining this with $\mathsf{E}[J] = \mathsf{E}[W] + \mathsf{E}[S]$ leads right away to[7]

[7] [5.4.7]

$$\mathsf{E}[J] = \frac{\mathsf{E}[S]}{1-\rho}, \qquad \mathsf{E}[L] = \frac{\rho}{1-\rho}, \qquad \mathsf{E}[Q] = \frac{\rho^2}{1-\rho}. \tag{5.4.9}$$

IN THE ABOVE PROOF of Little's law we assumed that there is an increasing sequence of epochs $\{T_k, k = 0, 1, \ldots\}$ at which the system is empty. However, in many practical queueing situations the system is never empty. In [5.4.9] and [5.4.10] we show that, for Little's law to holds, it suffices that the system is rate-stable.

Ex 5.4.1. Use the (physical) dimensions of the components of Little's law to check that $\mathsf{E}[J] \neq \lambda \mathsf{E}[L]$.

With this check, you can prevent making an often-made mistake.

Ex 5.4.2. Consider the server of the $G/G/1$ queue as a system by itself. Jobs arrive at rate λ and stay $\mathsf{E}[S]$ in this system. Use Little's law to conclude that $\rho = \lambda \mathsf{E}[S]$.

Ex 5.4.3. Compute $\mathsf{E}[W(M/M/1)]$ when $\lambda = 5/h$ and $\mu = 6/h$.

$\mathsf{E}[W(M/M/1)]$ *is the waiting time in queue for the* $M/M/1$ *queue*

Ex 5.4.4. Suppose that, at the moment you join the system of [5.4.3], the number of customers in the system is 10. What is your expected waiting time?

Ex 5.4.5. Derive (5.4.5).

Ex 5.4.6. Why is (5.4.8) *not* true in general for the $M/G/1$ queue?

Ex 5.4.7. Derive (5.4.9).

Ex 5.4.8. Explain that $\sum_{k=1}^{A(t)} J_k \geq \int_0^t L(s)\,\mathrm{d}s \geq \sum_{k=1}^{D(t)} J_k$.

Ex 5.4.9. Take suitable limits to show that $\lambda \mathsf{E}[J] \geq \mathsf{E}[L] \geq \delta \mathsf{E}[J]$. Where do you need the strong law of large numbers?

Continuation of [5.4.8]

Ex 5.4.10. Suppose that $A(t) = \lambda t$ and $D(t) = [A(t) - 10]^+$ so that the system is never empty for $t > 0$. Conclude that Little's law is still true.

5.5 GRAPHICAL SUMMARY

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

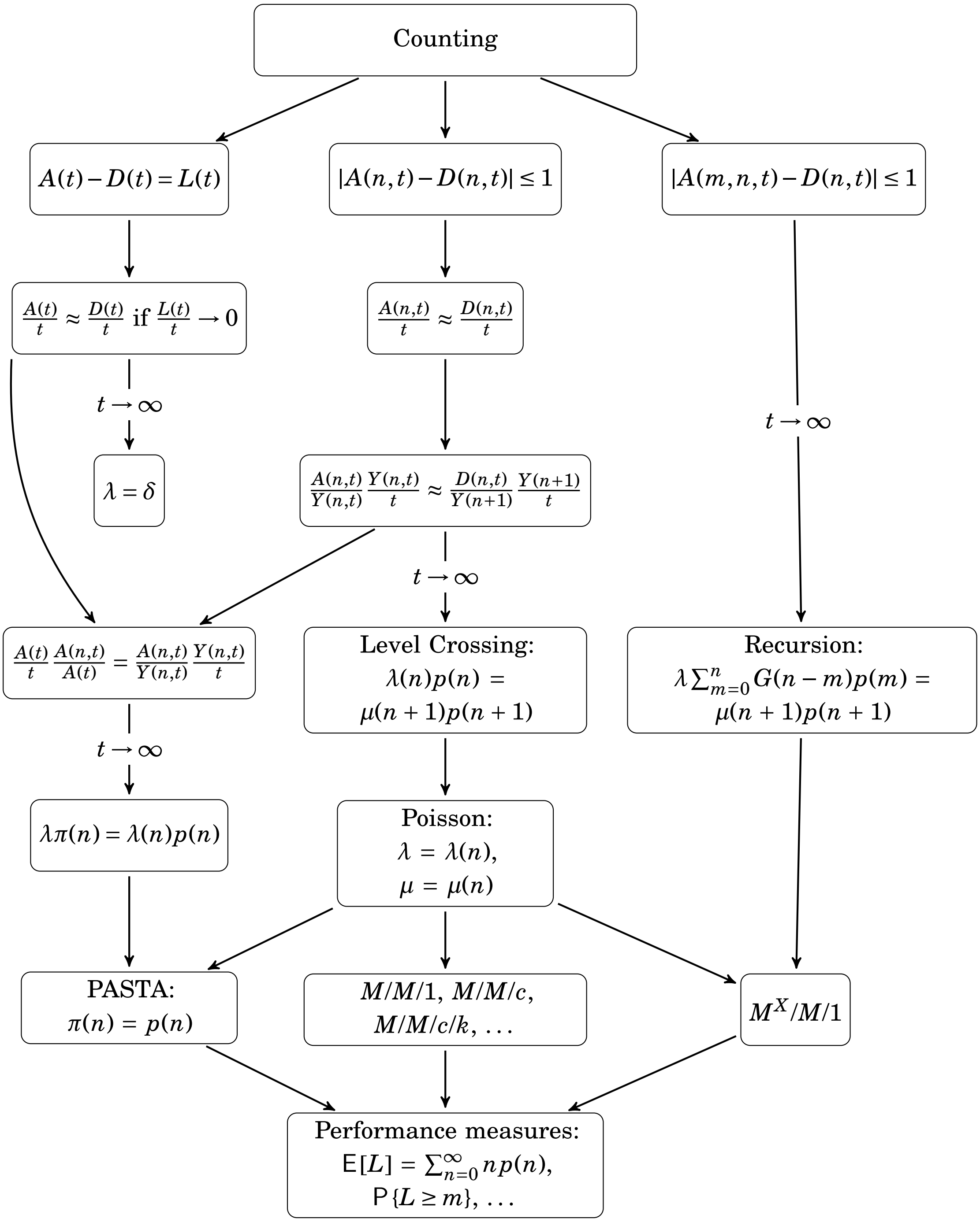


Figure 10: With level-crossing arguments we can derive a number of useful relations. This figure presents an overview of these relations that we derive in this and the next sections.

6 EXACT QUEUEING MODELS

In this chapter we use the concepts of Chapter 5 to model and analyze many queueing systems in steady state. The simplest, non-trivial, case is the $M/M/1$ queue, which is the topic of Section 6.1. As the main ideas are based on sample-paths, it turns out to be nearly trivial to extend the analysis of the $M/M/1$ to the $M(n)/M(n)/1$ queue in Section 6.2. The $M(n)/M(n)/1$ queue is a very generic model with which we can find closed-form expressions for the queue length distributions for many other queueing models such as the $M/M/c$ queue or the $M/M/1/K$, i.e., a queueing system with loss,

We then focus on finding the expected queueing time for batch queues, in Section 6.3, and the $M/G/1$ queue, in Section 6.4. In the last two sections, Section 6.5 and Section 6.6, we derive expressions for the queue length distributions of the batch queue and the $M/G/1$ queue.

Many of the queueing systems we analyze here are either generalizations of some other model, for instance, the $M(n)/M(n)/1$ generalizes to the $M/M/c$ queue, or reduce to special cases in certain parameter settings, such as that the $M/G/1$ becomes the $M/M/1$ queue when the service times are exponentially distributed. Quite a number of exercises in this chapter are targeted on *checking* that the general results reduce to those of the special cases. The reader should understand the importance of such checks. These exercises are simple in a sense—it is perfectly clear what to do, there is no model to make for instance—, but the algebra can be quite tough at times, and hence it is good practice.

6.1 $M/M/1$ QUEUE AND ITS VARIATIONS

With the level-crossing equations (5.2.10) and a judicious choice of $\lambda(n)$ and $\mu(n)$ we can model many different queueing systems. However, for the analysis we require that the time to the next arrival or departure is memoryless[0].

The exercises ask you to derive the formulas.[1] In Section 6.2 we show how to apply the models we derive here.

As said, in all models below the inter-arrival times and service times need to be memoryless. Specifically, we require that[2]

$$\mathsf{P}\{L(t+\Delta t) = n+1 \,|\, L(t) = n\} = \lambda(n)\Delta t + o(\Delta t), \quad n \geq 0, \tag{6.1.1}$$

$$\mathsf{P}\{L(t+\Delta t) = n-1 \,|\, L(t) = n\} = \mu(n)\Delta t + o(\Delta t), \quad n \geq 1. \tag{6.1.2}$$

In other words, the arrival rate $\lambda(n)$ and service rate $\mu(n)$ may depend on the number of jobs in the system, but not more.

WE TAKE $\lambda(n) = \lambda$ for the arrival rate and $\mu(n) = \mu$ for the service rate to model the $M/M/1$ queue[3]. Therefore, the level-crossing equations

[0] *hence, exponential*

[1] *The main challenge is not to make computational errors*

[2] *Compare the construction of the Poisson process in [2.2.5].*

[3] *$M/M/1$ queue*

become

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)} p(n) = \frac{\lambda}{\mu} p(n) = \rho p(n), \quad \rho = \frac{\lambda}{\mu}. \tag{6.1.3}$$

As this holds for any $n \geq 0$, $p(n+1) = \rho^{n+1} p(0)$. It follows from (5.2.12) and (1.2.3d) that $p(0) = 1-\rho$, hence

$$p(n) = (1-\rho)\rho^n, \quad n \geq 0. \tag{6.1.4}$$

It is now easy to compute the most important performance measures. The utilization is $\rho = \lambda/\mu$, and with a bit of algebra,[4]

$$\mathsf{E}[L_s] = \rho, \qquad \mathsf{E}[L] = \frac{\rho}{1-\rho}, \qquad \mathsf{P}\{L > n\} = \rho^{n+1}. \tag{6.1.5}$$

IF JOBS ARE BLOCKED[5] when $L \geq K$ we have the $M/M/1/K$ queue[6]. We can implement this by taking $\lambda(n) = \lambda \mathbb{1}_{n<K}$. This gives that $\lambda p(n) = \mu p(n+1)$ for $n < K$, and $p(n) = 0$ for $n > K$. Then, using (1.2.3d), it follows right away that

$$p(n) = \frac{1-\rho}{1-\rho^{K+1}} \rho^n, \quad \rho = \frac{\lambda}{\mu}. \tag{6.1.6}$$

Note that, contrary to the $M/M/1$ case,

$$\mathsf{E}[L_s] = \sum_{n=0}^{K} \min\{n,1\} p(n) = 1 - p(0) \neq \rho. \tag{6.1.7}$$

The rest of the performance measures follow easily. For instance, $\mathsf{E}[W] = \mathsf{E}[L]\,\mathsf{E}[S]$, because the 'average job' sees, upon arrival, $\mathsf{E}[L]$ in the system, so that the expected time to clear the queue is $\mathsf{E}[L]\,\mathsf{E}[S]$.

WE NEXT CONSIDER the $M/M/c$ queue[7]. For this, we set $\lambda(n) = \lambda$, and $\mu(n) = \mu \min\{n, c\}$. As in (4.1.1b), we take $\rho = \lambda/(c\mu)$. After some calculations we obtain[8]

$$p(n) = \frac{1}{G} \frac{1}{\prod_{k=1}^{n} \min\{c,k\}} (c\rho)^n, \tag{6.1.8a}$$

$$G = \sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!}, \tag{6.1.8b}$$

$$\mathsf{E}[Q] = \sum_{n=c}^{\infty} (n-c) p(n) = \frac{(c\rho)^c}{c!G} \frac{\rho}{(1-\rho)^2} \tag{6.1.8c}$$

$$\mathsf{E}[L_s] = \sum_{n=0}^{c} \min\{n,c\} p(n) = \frac{\lambda}{\mu}. \tag{6.1.8d}$$

Observe that the expected number of busy servers $\mathsf{E}[L_s]$ is equal to the load $\lambda\,\mathsf{E}[S] = \lambda/\mu$ (and not to the utilization ρ).

THE MULTI-SERVER QUEUE WITH BLOCKING, i.e., the $M/M/c/K$ queue,[9] is a simple generalization of the $M/M/c$ and $M/M/1/K$ queue. Now, $\lambda(n) = \lambda \mathbb{1}_{n<K}$ and $\mu(n) = \mu \min\{n,c\}$. The probabilities $p(n)$ are again given by (6.1.8a). The normalization is trivial, numerically at least. Here is an example in code.[10]

[4] [6.1.1]

[5] Or they are not willing to join the system

[6] $M/M/1/K$ queue

[7] $M/M/C$ queue

[8] [6.1.4]

[9] $M/M/C/K$ queue,

[10] The other models of this section are just as easily implemented. Note in particular that we directly use the recursion to compute $p(n)$, rather then the closed form expressions; using the former is numerically more efficient, at least for this case.

```
>>> import numpy as np

>>> labda, mu, c, K = 3, 2, 4, 8
>>> p = np.ones(K + 1, dtype=float)

>>> for n in range(K):
...     p[n + 1] = p[n] * labda / (mu * min(n + 1, c))
...
>>> p /= p.sum()  # normalize
>>> ELs = sum(p[n] * min(n, c) for n in range(len(p)))
```

THE $M/M/c/c$ QUEUE[11] is a special case of the $M/M/c/K$ queue: the number of servers is exactly equal to the number of jobs that is allows to enter the system. This queue system is also known as the Erlang B model and is often used to determine the number of beds needed by a hospital: the beds act as servers and the patients as jobs. Given the arrival rate of patients, and the average time they occupy a bed[12], the problem is to find c such that the loss probability $p(c)$ is less than 1%, say.

[11] *M/M/C/C queue*

[12] *i.e., the expected service time*

Assuming Poisson arrivals is reasonable since patients arrive independently from a large population.[13] Also, there are typically many patients in the hospital, hence the recovery times are quite accurately described by an exponential distribution.

[13] *And we can use PASTA to conclude that $\pi(c) = p(c)$, hence the loss fraction is indeed $p(c)$.*

Take $\lambda(n) = \lambda \mathbb{1}_{n<c}$, and $\mu(n) = n\mu$. As for the $M/M/c$ queue, $\rho = \lambda/(c\mu)$. Then some algebra[14] gives

[14] *[6.1.6]*

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!}, 0 \le n \le c, \qquad G = \sum_{n=0}^{c} \frac{(c\rho)^n}{n!}, \tag{6.1.9}$$

$$\mathsf{E}[Q] = 0, \qquad \mathsf{E}[L_s] = \frac{\lambda}{\mu}(1 - p(c)). \tag{6.1.10}$$

The last two results are easy. As there are as many servers as jobs, jobs cannot be in queue. Next, observing that $\lambda(1-p(c))$ is the rate of accepted jobs[15], the load is $\lambda(1-p(c))/\mu$, and this in turn is equal to the average number of busy servers.

[15] *Since, by PASTA, a fraction $p(c)$ is lost.*

THE AMPLE SERVER QUEUE, i.e., the $M/M/\infty$ queue[16], is simple. Any job that arrives finds a free server, hence, its service can start right away. Therefore, $\lambda(n) = \lambda$ and $\mu(n) = n\mu$ for all $n \ge 0$. By taking the limit $c \to \infty$ in the expressions of the $M/M/c$ queue (or the $M/M/c/c$ queue) we get

[16] *$M/M/\infty$ queue*

$$p(n) = e^{-\lambda/\mu}\frac{(\lambda/\mu)^n}{n!}, \qquad \mathsf{E}[L] = \mathsf{E}[L_s] = \frac{\lambda}{\mu}. \tag{6.1.11}$$

We see that the number of busy servers in the $M/M/\infty$ queue is Poisson distributed with parameter $\lambda\,\mathsf{E}[S]$. We mention in passing—but do not prove it—that the same results also hold for the $M/G/\infty$ queue.

IN QUEUES WITH A FINITE CALLING POPULATION[17] there is a fixed number, N say, of jobs. This model is useful to analyze repair systems and inventory systems of spare parts. For instance, consider a factory

[17] *Finite calling population*

with N machines and one mechanic.[18] A machine can be in one of two states: working or failed. When a machine breaks down, it moves to the repair department and waits until it is repaired. When n machines are in repair, there are $N-n$ machines still working. Thus, if λ is the rate at which a machine can fail, $\lambda(N-n)$ is the rate at which any of the working machines can fail. Since there is one mechanic: $\mu(n)=\mu$.

[18] When there are also N servers available, we obtain the Ehrenfest model of diffusion, which is used to explain the second law of thermodynamics.

With this model for $\lambda(n)$ and $\mu(n)$ the probabilities become [19]

[19] [6.1.8]

$$p(n) = \frac{N!}{(N-n)!}\frac{\rho^n}{G}, \qquad G = \sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!}. \tag{6.1.12}$$

As the expression for G cannot be simplified, there is not much point trying to derive simple expressions for $\mathsf{E}[L]$ and so on.

FINALLY, WE CONSIDER queues with balking,[20] 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 $\mu(n)=\mu$ and

[20] Balking,

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n=0, \\ \lambda/2, & \text{if } n=1, \\ 0, & \text{if } n>1. \end{cases} \tag{6.1.13}$$

Balking is not necessarily the same as blocking. In the latter case, $\lambda(n)=\lambda \mathbb{1}_{n<K}$, in the former case, a fraction of the customers may already choose not the join the system at a lower level than K.

As the steady-state probabilities depend entirely on the form of $\lambda(n)$ and $\mu(n)$, we have to use numerical methods to compute $p(n)$. With this, the arrival rate becomes $\lambda = \sum_{n=0}^{\infty}\lambda(n)p(n)$, and this is the arrival rate we need to use in, for instance, Little's law.[21]

[21] [6.2.4]

Ex 6.1.1. Use moment-generating functions to derive $\mathsf{E}[L]$ and $\mathsf{V}[L]$ for the $M/M/1$ queue, and show that $\mathsf{P}\{L>n\}=\rho^{n+1}$.

Ex 6.1.2. Numerically check that in [4.1.5], $\beta = 1-\mathsf{E}[L_s]\mu/\lambda$ for the $M/M/3/8$ queue with $\lambda=3$ and $\mu=2$.

Ex 6.1.3. Explain that for the $M/M/1$ queue $\mathsf{E}[Q]=\sum_{n=1}^{\infty}(n-1)\pi(n)$ and use this to find that $\mathsf{E}[Q]=\rho^2/(1-\rho)$.

Ex 6.1.4. Derive the expressions for the $M/M/c$ queue.

Ex 6.1.5. Check that the performance measures of the $M/M/c$ queue reduce to those of the $M/M/1$ queue if $c=1$.

Ex 6.1.6. Derive the expressions for the $M/M/c/c$ queue.

Ex 6.1.7. Derive the expressions for the $M/M/\infty$ queue.

Ex 6.1.8. Find the steady state probabilities for a single-server queue with a finite calling population with N jobs.

Ex 6.1.9. Derive the steady state probabilities $p(n)$ for a queue with a finite calling population with N jobs and N servers. What happens if $N\to\infty$?

Ex 6.1.10. Show that as $K \to \infty$, the performance measures of the $M/M/1/K$ converge to those of the $M/M/1$ queue.

Ex 6.1.11. Derive the expression for $\mathsf{E}[L]$ in (6.1.5) by means of indicator variables.

Ex 6.1.12. Derive $\mathsf{E}[L]$ and $\mathsf{E}\left[L^2\right]$ by differentiating the LHS and RHS of $\sum_{n=0}^{\infty} \rho^n = (1-\rho)^{-1}$.

6.2 APPLICATIONS OF LEVEL-CROSSING, LITTLE'S LAW AND PASTA

In this section we apply the tools developed up to now to many different queueing situations. As an example, we discuss in considerable detail how to plan the number of cashiers at a supermarket such that the queue length remains small. To keep the analysis simple, we approximate the $M/M/c$ queue by an $M/M/1$ with a server that is c times as fast. So we first address the quality of this approximation.

The exercises at then end require quite some modeling skills. Hence, they are quite hard, even thought the formulas are simple.

It should be clear from Section 6.1 that the expressions for, e.g., $\mathsf{E}[L]$ are quite a bit harder for the $M/M/c$ queue than for the $M/M/1$. To avoid using this complexity, it is tempting to approximate the $M/M/c$ queue by an $M/M/1$ queue with a server that works c times as fast.

Let us therefore consider a numerical example. Suppose that we have an $M/M/3$ queue with $\lambda = 5$ per day and $\mu = 2$ per day per server, and we compare it to an $M/M/1$ with the same arrival rate but with a service rate of $\mu = 3 \cdot 2 = 6$.

The figure at the right shows the ratio of $\mathsf{E}[J]$ for both queues as a function of ρ. Clearly, when ρ is small, this ratio is about 3. This is 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, and when ρ is small, $\mathsf{E}[J] \approx \mathsf{E}[S]$ because $\mathsf{E}[Q] \approx 0$. However, when ρ is large, $\mathsf{E}[Q] \gg \mathsf{E}[S]$. Moreover, jobs in queue do not 'see' the difference between c slow servers or one fast server. Therefore, the ratio becomes 1 as $\rho \uparrow 1$.

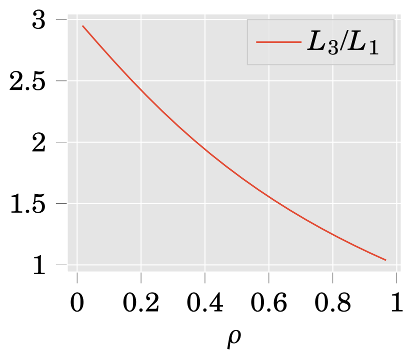


Figure 11: The ratio of $\mathsf{E}[L]$ for the $M/M/3$ and $M/M/1$ queue.

LET US NOW DISCUSS THE example of cashier planning of a supermarket. We keep the analysis simple, as otherwise we would need sophisticated queueing models and data assembly tools. Note however that the present example contains all necessary steps to solve realistic planning problems.

We start with the arrival process distribution. It is reasonable to model it as a Poisson process, as there are many potential customers, each choosing with a small probability to go to the supermarket at a certain moment in time. Thus, we only have to characterize the arrival rate. Estimating this for a supermarket is easy because the cash registers track all customers payments. With this we have the number of customers that left, hence arrived at, the shop.

It is common to aggregate this data in a *demand profile* which shows the average number of customers arriving per hour. Then we model the

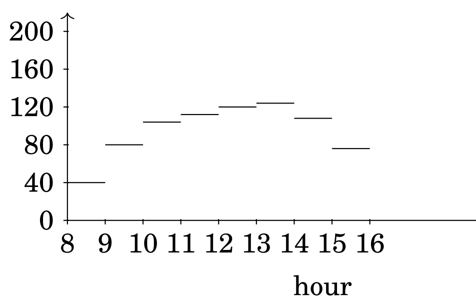


Figure 12: A demand profile.

arrival process as Poisson with an arrival rate that is constant during a certain hour as specified by the demand profile.

It is also easy to find the service distribution from the cash registers. The first item scanned after a payment determines the start of a new service, and the payment closes the service.[0] To keep things simple here, we just model the service time distribution as exponential with mean 1.5 minutes.

[0] *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.*

For the *service objective* we consider the number of people in entire queueing system. For ease, suppose there are 5 cashier positions, and that the queue per position should be at most 3. Then there can be $5 \cdot (1+3) = 20$ in the system before we say that the system is 'too full'. As objective, let us set $\mathsf{P}\{L > 20\} < 1\%$.

Let us model the situation as an $M/M/1$ queue with a fast server, so that the objective becomes easy to compute. In fact, since $\mathsf{P}\{L > n\} = \rho^{n+1}$, we obtain the constraint $\rho \leq 0.8$. In view of Fig. 11, the difference in $\mathsf{E}[L]$ between the multi-server and the single, but fast, server queue is small.

We now find a formula to convert the demand profile into a *load profile* which specifies the minimal number of servers per hour needed to meet the service objective. Using that $\rho = \lambda \mathsf{E}[S]/c \leq 0.8$ and our estimate $\mathsf{E}[S] = 1.5 \cdot 60$ hour, we get the following rough bound on c:

$$c \geq \lambda \frac{\mathsf{E}[S]}{0.8} \approx 0.03\lambda. \tag{6.2.1}$$

For instance, for the hour from 12 to 13, we read in the demand profile Fig. 12 that $\lambda = 120$ customers per hour, hence $c \approx 3.8 = 4$.

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. For instance, a shift length must be at least four hours, and no longer than 9 hours including breaks. Or, when the shift is longer than 4 hours it needs to contain at least one break of 30 minutes. Shifts can also have different costs: evening shifts are typically more expensive per hour.

The usual way to solve such covering problems is by means of an integer problem. For instance, suppose only 4 *shift plans* are available as shown at the right.[1] Then generate *shift types* where a shift type is a shift plan that starts at a certain hour.

[1]

1. $++-++$
2. $+++-+$
3. $++-+++$
4. $+++-++$,

Shift plans. A + indicates a working hour and − a break of an hour.

For the optimization, let x_i be the number of shifts of type i and c_i the cost of this type. Then the problem is to solve $\min \sum_i c_i x_i$, such that for all hours t the shifts cover the load, i.e., $\sum_i x_i \mathbb{1}_{t \in s_i} \geq 0.03\lambda_t$. (We write $t \in s_i$ if hour t is covered by shift type i.)

Ex 6.2.1 (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 $\mathsf{E}[Q]$, $\mathsf{E}[L]$ 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 (waiting and in service) customers can sit down at least 90 percent of the time. How many chairs should he provide?

Ex 6.2.2 (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. Determine $\mathsf{E}[L]$, $\mathsf{E}[Q]$, $\mathsf{E}[J]$, $\mathsf{E}[W]$, $\mathsf{V}[L]$ and $\mathsf{V}[Q]$. Finally, compute the service time and the utilization.

Ex 6.2.3. (Hall 5.14) An airline phone reservation line has one server and a buffer for two customers. The arrival rate is 6 customers per hour, and a service rate of just 5 customers per hour. Arrivals are Poisson and service times are exponential. Estimate $\mathsf{E}[Q]$ and the average number of customers served per hour. Then, estimate $\mathsf{E}[Q]$ for a buffer of size 5. What is the impact of the increased buffer size on the number of customers served per hour?

Ex 6.2.4 (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 $\mathsf{E}[L]$, $\mathsf{E}[Q]$, $\mathsf{E}[J]$ and $\mathsf{E}[J]$.

Ex 6.2.5 (Hall 5.10)**.** A repair/maintenance facility would like to determine how many employees should be working in its tool crib. The customers are actually maintenance workers at the facility, and are compensated at the same rate as the tool crib employees. The service time is exponential, with mean 4 minutes, and customers arrive by a Poisson process with rate 28 per hour. What is $\mathsf{E}[W]$ for $c = 1, 2, 3$, or 4 servers? How many employees should work in the tool crib?

Recall, in queueing systems always somebody has to wait, either a customer in queue or a 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.

Ex 6.2.6 (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). First find an appropriate way to model this queueing system as an $M/M/1$ queue. Then calculate the expected number of cabs waiting and the expected number of parties waiting, and the average waiting times of cabs and parties. What would be the impact of allowing four cabs to wait at a time? 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.

This is a bit hard, but important problem; it is the same as an inventory system with backlogging in which taxis act as products on hand and parties as customers.

Ex 6.2.7. Suppose cabs are not allowed to wait. What is the expected waiting time for a party of riders?

Continuation of [6.2.6]

Ex 6.2.8. Suppose cabs can contain at most 4 riders, and the size of a party (i.e., a batch) has distribution B_k with $\mathsf{P}\{B_k = i\} = 1/7$ for $i =$

Continuation of [6.2.6]. This one is quite hard.

$1,\dots,7$. Parties of riders have the same destination, so riders of different parties cannot be served by one taxi. Provide a set of recursions to simulate this system.

6.3 $M^X/M/1$ QUEUE: EXPECTED WAITING TIME

Sometimes jobs arrive in batches, rather than as single units. For instance, when a car or a bus arrives at a fast-food restaurant, a batch consists of the number of people in the vehicle. When the batches arrive as a Poisson process and the individual items within a batch have exponential service times we denote such queueing systems by the shorthand $M^X/M/1$. For this queueing model we derive expressions for the load, and the expected waiting time and number of jobs in the system.

We will use the renewal-reward theorem in two ways. In the first we focus on entire batches at the server, in the other on single items.

ASSUME THAT JOBS arrive as a Poisson process with rate λ and each *job* contains multiple *items*. Denote by B_k the batch size of the kth job. We assume that $\{B_k\}$ is a sequence of i.i.d. discrete random variables whose common distribution is that of B. The service time of each item is $\mathsf{E}[S]$. The utilization is therefore $\rho = \lambda \mathsf{E}[B]\mathsf{E}[S]$. We require that $\rho < 1$.

SUPPOSE THAT AN arriving batch joins the end of the queue (if present), and once the queue in front of it has been cleared, it moves in its entirety to the server.[0] Thus, all items in one batch spend the same time in queue. Once the batch moves to the server, the server processes the items one after another until the batch is finished. Write Q^B for the number of batches in queue and L_s^B for the number of items of the job (if any) at the server.

[0] *This is the same batching model as in Section 4.2.*

Observe that the average time a batch spends in queue is

$$\begin{aligned}\mathsf{E}\left[W^B\right] &= \mathsf{E}\left[Q^B\right]\mathsf{E}[B]\,\mathsf{E}[S] + \mathsf{E}\left[L_s^B\right]\mathsf{E}[S] &(6.3.1)\\ &= \mathsf{E}[L]\,\mathsf{E}[S] = \mathsf{E}[W]. &(6.3.2)\end{aligned}$$

Hence, $\mathsf{E}\left[W^B\right]$ is equal to the average time an item spends in queue.[1] Next, replacing in the above $\mathsf{E}\left[Q^B\right]$ by[2] $\lambda\mathsf{E}\left[W^B\right]$, we obtain

[1] *As it should.*

[2] *Little's law*

$$\mathsf{E}\left[W^B\right] = \frac{\mathsf{E}\left[L_s^B\right]}{1-\rho}\mathsf{E}[S], \qquad \mathsf{E}[L] = \frac{\mathsf{E}\left[L_s^B\right]}{1-\rho}. \tag{6.3.3}$$

So let us find an expression for $\mathsf{E}\left[L_s^B\right]$.

FOR THIS WE can use the renewal reward theorem. With [5.1.1] as inspiration,[3] define on the one hand $Y(t) = \int_0^t L_s^B(s)\,\mathrm{d}s$ as the time spent by the items at the server. Then, $\lim_{t\to\infty} Y(t)/t = Y = \mathsf{E}\left[L_s^B\right]$. On the other hand, with D_k the time the kth job leaves the server, $X_k = Y(D_k) - Y(D_{k-1})$ is the time spent by the items of this job at the server. Then, a bit of work shows that[4]

[3] *Solve this exercise if you have not done yet.*

[4] *[6.3.4]*

$$\mathsf{E}[X] = \frac{\mathsf{E}\left[B^2\right] + \mathsf{E}[B]}{2}\mathsf{E}[S]. \tag{6.3.4}$$

Since $Y = \lambda X$, [5]

[5] [6.3.5]

$$\mathsf{E}\left[L_s^B\right] = \lambda \frac{\mathsf{E}\left[B^2\right]}{2} \mathsf{E}[S] + \frac{\rho}{2}. \tag{6.3.5}$$

Finally, with a bit of polishing[6] we can rewrite this into

[6] [6.3.6]

$$\mathsf{E}[L] = \frac{1+C_s^2}{2} \frac{\rho}{1-\rho} \mathsf{E}[B] + \frac{1}{2} \frac{\rho}{1-\rho}, \tag{6.3.6}$$

where C_s^2 is the SCV of the batch size distribution.

RATHER THAN USING the cost of an entire batch spending time at the server, as in the above use of the renewal-reward equation, we can also concentrate on the cost of single items at the server. This will provide us with a simple expression for $\mathsf{P}\{L_s = i\}$, which in turn leads to another derivation of (6.3.5).

If $L_s(t)$ is the number of items (of the batch in service) at the server at time t, then $Y_i(t) = \int_0^t \mathbb{1}_{L_s(s)=i}\, ds$ is the total time there are i items at the server. From Fig. 13 it follows that $\int_{D_{k-1}}^{D_k} \mathbb{1}_{L_s(s)=i}\, ds = \mathbb{1}_{B_k \geq i} S_{k,i}$, where $S_{k,i}$ is the service time of the ith item of this batch. But then $Y_i(D_n) = \sum_{k=1}^n \mathbb{1}_{B_k \geq i} S_{k,i}$.

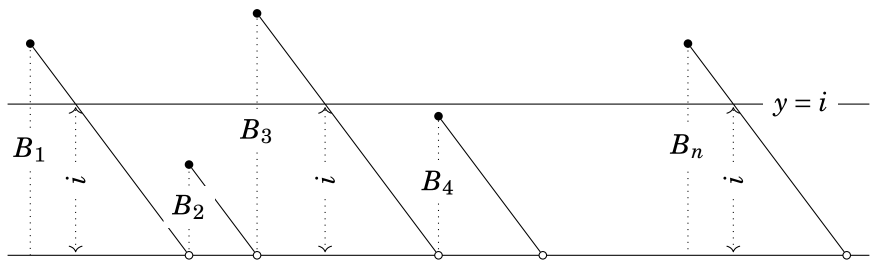


Figure 13: A batch crosses the line $y = i$ iff. it contains at least i items. Thus, during the service of a batch with i or more items, there is precisely one service of an i-th item.

By sampling $Y(\cdot)$ at departure times $\{D_k\}$, we see that $X_k = Y_i(D_k) - Y_i(D_{k-1}) = S_{k,i}\,\mathbb{1}_{B_k \geq i}$. Since the $\{S_{k,i}\}$ are i.i.d. with $\mathsf{E}\left[S_{k,i}\right] = \mathsf{E}[S]$, and the $\{B_k\}$ are i.i.d., we obtain

$$X = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^n S_{k,i}\,\mathbb{1}_{B_k \geq i} = \mathsf{E}[S\,\mathbb{1}_{B \geq i}]. \tag{6.3.7}$$

But B and S are independent by assumption, hence $X = \mathsf{E}[S]\,\mathsf{E}[\mathbb{1}_{B\geq i}] = \mathsf{E}[S] G(i-1)$, since G is the survivor function of B. As for Y, by construction, $Y_i(t)/t \to \mathsf{P}\{L_s = i\}$. Using the renewal reward theorem and rate stability ($\delta = \lambda$) we conclude that

$$\mathsf{P}\{L_s = i\} = \lambda\,\mathsf{E}[S] G(i-1) = \rho \frac{G(i-1)}{\mathsf{E}[B]}. \tag{6.3.8}$$

Now that we have $\mathsf{P}\{L_s = i\}$,

$$\mathsf{E}[L_s] = \sum_{i=0}^{\infty} i\,\mathsf{P}\{L_s = i\} = \frac{\rho}{\mathsf{E}[B]} \sum_{i=1}^{\infty} i G(i-1). \tag{6.3.9}$$

Simplifying[7] the summation yields again (6.3.6).

[7] [6.3.7]

Ex 6.3.1. 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 $\mathsf{P}\{B=1\} = \mathsf{P}\{B=2\} = \mathsf{P}\{B=3\} = 1/3$. Batches arrive at rate $\lambda = 1/h$. The average processing time for an item is 25 minutes. Compute by how much $\mathsf{E}[L]$ would decrease if $B \equiv 2$.

Of course, it is up to management to decide whether such reductions outweigh any efforts to reduce the variation in batch sizes.

Ex 6.3.2. Show that $\mathsf{E}\left[W(M^X/M/1)\right] \geq \mathsf{E}[W(M/M/1)]$ even when the loads are the same. What do you conclude?

Relate this to Sakasegawa's formula.

Ex 6.3.3. Compute $\mathsf{E}[L]$ when B is geometrically distributed with $\mathsf{E}[B] = 1/p$. Check that if $\mathsf{E}[B] = 1$ the $M/M/1$ queue results.

Ex 6.3.4. Show (6.3.4).

This exercise uses similar tools as in Section 4.4.

Ex 6.3.5. Complete the argumentation that leads to (6.3.5).

Ex 6.3.6. Show that $\mathsf{E}\left[B^2\right]/(\mathsf{E}[B])^2 = 1 + C_s^2$.

Ex 6.3.7. Show that $\sum_{i=1}^\infty iG(i-1) = (\mathsf{E}\left[B^2\right] + \mathsf{E}[B])/2$.

6.4 $M/G/1$ QUEUE: EXPECTED WAITING TIME

When the service times are not really well approximated by the exponential distribution, the $M/G/1$ queue may be a more suitable better model than the $M/M/1$ queue. In this section we derive with sample path arguments the fundamentally important Pollaczek-Khinchine (PK) formula for the average waiting time in an $M/G/1$ queue. At the end we relate Sakasegawa's formula to the PK formula.

WE START WITH the simple observation an arriving job first has to wait until the job in service (if any) completes and then has wait for the queue to be served. From PASTA it then follows that $\mathsf{E}[W] = \mathsf{E}[S_r] + \mathsf{E}[Q]\,\mathsf{E}[S]$, where $\mathsf{E}[S_r]$ is the (time-)average remaining service time of the job in service. With Little's law $\mathsf{E}[Q] = \lambda\,\mathsf{E}[W]$ we can rewrite this into

$$\mathsf{E}[W] = \frac{\mathsf{E}[S_r]}{1-\rho}. \tag{6.4.1}$$

TO MAKE FURTHER progress we need to find an expression for $\mathsf{E}[S_r]$ for generally distributed service times. For this, we can use the renewal reward theorem, just as in Section 6.3.

Consider the kth job of some sample path of the $M/G/1$ queueing process. Let its service time start at time $\tilde{A}_k$ so that it departs at time $D_k = \tilde{A}_k + S_k$, see Fig. 14. At time s, the remaining service time of job k is $(D_k - s)\mathbb{1}_{\tilde{A}_k \leq s < D_k}$. By adding up all these times, we find that[0]

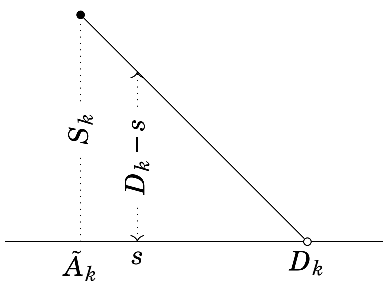


Figure 14: Remaining service time.

[0] [6.4.6]

$$Y(t) = \int_0^t (D_{D(s)+1} - s)\mathbb{1}_{L(s)>0}\,\mathrm{d}s \tag{6.4.2}$$

is the total remaining service time as seen by the server up to t. Hence, as $t \to \infty$, $Y(t)/t$ converges to the (time-average) remaining service time $\mathsf{E}[S_r]$.

We also see in Fig. 14 that $X_k = Y(D_k) - Y(D_{k-1})$ is the area under the triangle. By choosing $T_k = D_k$ as the epochs to inspect $Y(\cdot)$, $X = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^n S_k^2/2 = \mathsf{E}\left[S^2\right]/2$. By the renewal-reward theorem, $Y = \delta X$, but $\delta = \lambda$,[1] and therefore

$$\mathsf{E}[S_r] = \frac{\lambda}{2}\mathsf{E}\left[S^2\right]. \tag{6.4.3}$$

[1] by rate-stability

Replacing the above expression in the above expression for $\mathsf{E}[W]$, we obtain the Pollaczek-Khinchine formula[2]

[2] Pollaczek-Khinchine formula

$$\mathsf{E}[W] = \frac{1}{2}\frac{\lambda\mathsf{E}\left[S^2\right]}{1-\rho} = \frac{1+C_s^2}{2}\frac{\rho}{1-\rho}\mathsf{E}[S], \tag{6.4.4}$$

where the second equation follows after some rewriting.[3]

[3] [6.4.7]

THE PK FORMULA is an exact result for $\mathsf{E}[W]$, and it reduces to $\mathsf{E}[W(M/M/1)]$ by taking[4] $C_s^2 = 1$ in the factor $(1+C_s^2)/2$. In a similar vein, Sakasegawa's formula for the $G/G/c$ queue reduces to the PK formula for $\mathsf{E}[W(M/G/1)]$ when the number of servers $c = 1$ and setting[5] $C_a^2 = 1$ in the factor $(C_a^2 + C_s^2)/2$. Note, again, that Sakasegawa's formula is an approximation.

[4] *Recall, $C_s^2 = 1$ for the $M/M/1$ queue.*

[5] *Recall, $C_a^2 = 1$ for the $M/G/1$ queue.*

Ex 6.4.1. Consider a workstation with just one machine. We model the job arrival process as a Poisson process with rate $\lambda = 3$ per day. The average service time $\mathsf{E}[S] = 2$ hours, $C_s^2 = 1/2$, and the shop is open for 8 hours per day. Show that $\mathsf{E}[W] = 4.5h$. What would you propose to reduce $\mathsf{E}[W]$ to 2h?

Ex 6.4.2. Compare $\mathsf{E}[W(M/D/1)]$ to $\mathsf{E}[W(M/M/1)]$.

Ex 6.4.3. Compute $\mathsf{E}[J]$ for the $M/G/1$ queue with $S \sim U[0,\alpha]$.

Ex 6.4.4. Find an expression in terms of λ and $\mathsf{E}[S]$ for the acceptance and loss probabilities of the $M/G/1/1$ queue. Why is this expression not valid for the $G/G/1/1$ queue.

Ex 6.4.5. Show that for the $M/G/1$ queue, the expected idle time $\mathsf{E}[I] = 1/\lambda$ and the expected busy time $\mathsf{E}[U] = \mathsf{E}[S]/(1-\rho)$.

Ex 6.4.6. Explain the expression for $Y(t)$.

Ex 6.4.7. Complete the algebra in (6.4.4).

Ex 6.4.8. Show for the $M/G/1$ that $\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \mid S_r > 0]$.

Ex 6.4.9. It is an easy mistake to think that $\mathsf{E}[S_r] = \mathsf{E}[S]$ when service times are exponential. Why is this wrong?

Ex 6.4.10. Show for the $M/G/1$ that with probability ρ a job leaves behind a busy server.

6.5 $M^X/M/1$ QUEUE LENGTH DISTRIBUTION

In Sections 6.3 and 6.4 we established the Pollaczek-Khinchine formula for the waiting times of the $M^X/M/1$ queue and the $M/G/1$ queue, respectively. To compute more difficult performance measures such as the loss probability $\mathsf{P}\{L > n\}$, we need expressions for the stationary distribution $\pi(n)$. Here we present a numerical, recursive, scheme to compute these probabilities.

WE CAN AGAIN use level-crossing arguments to find $\{\pi(n)\}$. However, the reasoning that led to the level-crossing equation (5.2.6) needs to be generalized because, when a multi-item batch arrives, multiple levels will be crossed, see Fig. 15.

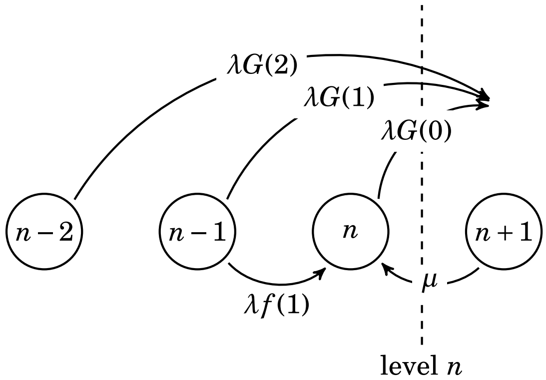


Figure 15: When $L(t) = n$, the arrival of any batch crosses level n from below, but when $L(t) = n-1$, only a batch larger than 1 ensures a crossing of level n, and so on.

Let us first guess what could be the appropriate generalization of $\lambda\pi(n) = \mu\pi(n+1)$.[0] The down-crossings are easy: since items are served one-by-one, we still must have $\mu\pi(n+1)$. For the upcrossings, if we only focus on the arrivals that see $m \leq n$ in the system, we see a thinned stream[1] with rate $\lambda\pi(m)$. But, by PASTA, the arrival process $\{N(u), u \geq t\}$ is independent of $\{L(s), s < t\}$, hence the thinned process is also Poisson. To cross level n, the batch size of an arriving job must be $> n-m$. Thinning once again with probability $G(n-m)$, and noting that the batch sizes are independent of the arrival times, we obtain yet again a Poisson process, but with rate $\lambda\pi(m)G(n-m)$. We can also merge[2] all these thinned Poisson processes, and form a Poisson process with rate $\lambda\sum_{m=0}^{n}\pi(m)G(n-m)$. By level-crossing, the up-crossing and down-crossing rates must match, and therefore,

$$\lambda \sum_{m=0}^{n} \pi(m)G(n-m) = \mu\pi(n+1). \tag{6.5.1}$$

It is easy to see that (6.5.1) holds for the $M/M/1$ queue.[3] Moreover, we can use it to derive $\mathsf{E}[L]$.[4]

WE CAN FORMALIZE the above as follows. Define

$$A(m,n,t) = \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=m}\,\mathbb{1}_{B_k>n-m} \tag{6.5.2}$$

[0] i.e., the level-crossing equations of the $M/M/1$ queue.

[1] Compare [2.2.14] and [2.2.17].

[2] Compare [2.2.12] and [2.2.18].

[3] [6.5.1]

[4] [6.5.2]

as the number of jobs up to time t that see m in the system upon arrival and have batch size larger than $n-m$. From Fig. 15 we see that $A(n,m,t)$ counts the number of up-crossings of level n up to time t.

As earlier, we are interested in the limit $A(n,m,t)/t$ as $t \to \infty$. If we follow the reasoning of Section 5.3, we obtain by multiplying and dividing,

$$\frac{A(m,n,t)}{t} = \frac{A(t)}{t}\frac{A(m,t)}{A(t)}\frac{A(m,n,t)}{A(m,t)}. \tag{6.5.3}$$

We already know that $A(t)/t \to \lambda$ and $A(m,t)/A(t) \to \pi(m)$; so, let us provide an interpretation for the third term.

If we fill in the definitions, we get

$$\frac{A(m,n,t)}{A(m,t)} = \frac{\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=m, B_k>n-m}}{\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=m}}. \tag{6.5.4}$$

In words, this is the fraction of those jobs that see m in the system upon arrival and have size $> n-m$. But recall that we assumed that $\{B_k\}$ are independent of what jobs see upon arrival. Thus, we can just as well count all jobs of size $> n-m$ along a sample path instead of those that see m upon arrival; in the limit the statistics must be the same. Therefore

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=m, B_k>n-m}}{\sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=m}} = \lim_{t\to\infty} \frac{\sum_{k=1}^{A(t)} \mathbb{1}_{B_k>n-m}}{A(t)} = G(n-m). \tag{6.5.5}$$

By level-crossing, the number of up- and down-crossing must satisfy

$$\left| \sum_{m=0}^{n} A(m,n,t) - D(n,t) \right| \leq 1. \tag{6.5.6}$$

Then, by combining the above, dividing by t, and taking the limit $t \to \infty$, we obtain (6.5.3).

IT IS LEFT to find the normalization constant. As (6.5.1) does not lead to a closed form expression for $\pi(n)$, such as (6.1.4), we have to specify a criterion to stop this iterative procedure. We discuss three different strategies to cope with this problem.

For the actual computations we suppose henceforth that $B \leq N$ for some N. Then, as $G(n) = 0$ for $n \geq N$, we can simplify (6.5.1) to

$$\mu\pi(n+1) = \lambda \sum_{m=0}^{\min\{n,N-1\}} \pi(n-m)G(m). \tag{6.5.7}$$

PERHAPS THE SIMPLEST strategy is to 'ignore the problem'. Just set $\pi(0) = 1$ to start the iteration, then compute the first M terms, for some M, and take $\sum_{i=0}^{M} \pi(i)$ as the normalization constant. With this, $\mathsf{E}[L] \approx \sum_{n=0}^{M} n\pi(n)$.

```python
import numpy as np


```

```python
def compute_pi(f, M):
    pi = np.ones(M + 1)
    F = np.cumsum(f)
    G = np.ones_like(F) - F
    N = len(G)

    for n in range(M):
        pi[n + 1] = sum(pi[n - m] * G[m] for m in range(min(n + 1, N)))
        pi[n + 1] *= labda / mu
    return pi / pi.sum()

labda, mu = 1, 3
f = np.array([0, 1, 1, 1])
f = f / f.sum()

pi = compute_pi(f, M=10)
EL = sum(n * pi[n] for n in range(len(pi)))
```

When running this code, we get $\mathsf{E}[L] = 2.501$.

But is $M = 10$ large enough? For this, we can compare $\mathsf{E}[L]$ as found above with (6.3.6).

```python
EB = sum(k * fk for k, fk in enumerate(f))
rho = labda * EB / mu
EB2 = sum(k * k * fk for k, fk in enumerate(f))
VB = EB2 - EB * EB
C2 = VB / EB / EB
EL_exact = (1 + C2) / 2 * rho / (1 - rho) * EB + rho / (1 - rho) / 2
```

Now we find for the exact value $\mathsf{E}[L] = 3.333$. So, simply stopping at some arbitrary value is not a viable procedure.

A better criterion is perhaps to stop when $\pi(M) \ll 1$.[5]

```python
1  def compute_pi_2(f, eps):
2      F = np.cumsum(f)
3      G = np.ones_like(F) - F
4      N = len(G)
5
6      pi, n = {}, 0
7      pi[n] = 1
8
9      while pi[n] > eps:
10         pi[n + 1] = sum(pi[n - m] * G[m] for m in range(min(n + 1, N)))
11         pi[n + 1] *= labda / mu
12         n += 1
13
14     norm = sum(pi[n] for n in pi.keys())
15     return {n: pi[n] / norm for n in pi.keys()}
```

[5] *In line 6 we choose to implement p as a dictionary, because we do not yet know how many terms it needs to have before satisfying the stopping criterion.*

```python


pi = compute_pi_2(f, eps=0.001)
EL = sum(n * pi[n] for n in pi.keys())
```

Now we find $\mathsf{E}[L] = 3.304$. This is indeed better, but $\epsilon = 0.001$ is not yet small enough: the second digit is wrong.[6]

[6] *Observe how important it is to have exact results such as (6.3.6) to check the results of numerical algorithms.*

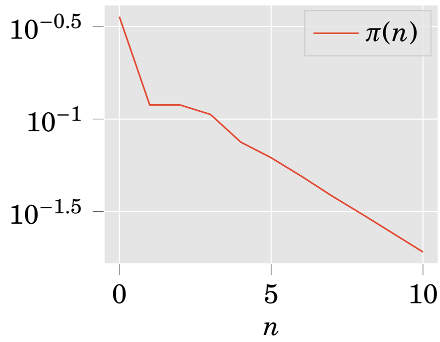
THE SECOND METHOD is exact, but more involved: we simply block demand above a certain level. There are three common policies to decide which items in a batch to accept.

1. Complete acceptance: accept all batches that arrive when the system contains K or fewer items, and reject the entire batch otherwise.[7]
2. Partial acceptance: accept whatever fits of a batch, and reject the rest.[8]
3. Complete rejection: if a batch does not fit entirely into the system, it will be rejected completely.[9]

[7] *[6.5.5]*

[8] *[6.5.4]*

[9] *[6.5.6]*



Once we have recursions similar to (6.5.1) we can again obtain $\{\pi(n)\}$.

THE THIRD METHOD relies on the fact that the numbers $\pi(n)$ decrease geometrically fast for n sufficiently large. Let us check this for the numerical example we studied above. In the figure at the right we plot $\log(\pi(n))$ as a function of n. Clearly, $\pi(n), n \geq 5$ seems to decrease as a straight line.

To provide some intuition for this fact, observe that for $N < n \leq 2N-1$,

$$\pi(n+1) = \frac{\lambda}{\mu}\sum_{m=0}^{N-1}\pi(n-m)G(m) < \frac{\lambda}{\mu}\sum_{m=0}^{N-1}G(m) = \rho < 1. \tag{6.5.8}$$

as $\pi(n) < 1$ for $n \leq N$. But then, for $2N < n \leq 3N-1$,

$$\pi(n+1) = \frac{\lambda}{\mu}\sum_{m=0}^{N-1}\pi(n-m)G(m) < \frac{\lambda}{\mu}\sum_{m=0}^{N-1}G(m)\rho = \rho^2, \tag{6.5.9}$$

and so on. However, this is a crude upper bound for the decay rate, because $\pi(n) < 1$ is not very sharp.

We can do better by substituting the form $\pi(n) = C\alpha^n$ into (6.5.1) for $k > N$. Then α should be a root of the equation $\lambda\sum_{m=0}^{N-1}\alpha^m G(N-1-m) = \mu\alpha^N$. Now this is a polynomial (in α) of degree N, which has N roots in general. We leave it to the interested reader to study the theoretical ramifications of this method, for instance, why should α be the largest root?

Ex 6.5.1. Show that (6.5.1) reduces to $\lambda\pi(n) = \mu\pi(n+1)$ for the $M/M/1$ case.

Ex 6.5.2. Use (6.5.1) in $\mathsf{E}[L] = \sum_{n=0}^{\infty} n\pi(n)$ to derive (6.3.6).

This is an important check both on (6.3.6) and (6.5.1).

Ex 6.5.3. Why is $\pi(n) = \delta(n)$ not true for the $M^X/M/1$ queue?

Ex 6.5.4. Derive a set of recursions analogous to (6.5.1) to compute $\pi(n)$ for the $M^X/M/1/K$ queue with partial acceptance.

Ex 6.5.5. Derive a set of recursions for the $M^X/M/1/K$ queue with complete acceptance.

Ex 6.5.6. Derive a set of recursions for the $M^X/M/1/K$ queue with complete rejection.

6.6 $M/G/1$ QUEUE LENGTH DISTRIBUTION

In Section 6.5 we used level-crossing arguments to find a recursive for the stationary distribution $\pi(n)$ of the $M^X/M/1$ queue. Here we find a similar recursion to compute $\pi(n)$ of $M/G/1$ queue. However, we cannot simply copy the ideas of Section 6.5 to the present situation, 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.

IF WE WANT to characterize the state at all moments in time, we need to keep track of the number of jobs in the system *and* the remaining service time of the job in service (if any), because service times are not memoryless. But, by sampling at job departure times $\{D_k\}$,[0] we can restrict the state description to just the number in the system. We say that the system is in state n at time D_k when $L(D_k) = n$.[1]

Let Y_k denote the number of arrivals during the service time of job k. Note that, because the service times are i.i.d., $\{Y_k\}$ is a sequence of i.i.d. random variables. Let Y be the common random variable with (pmf) $f(j) = \mathsf{P}\{Y = j\}$ and G as survivor function.

LET US CONCENTRATE on the down-crossing rate of level n.[2] Suppose we start the service of job k when the system is in state $n+1$.[3] When $Y_k = 0$, the system contains one job less after the departure of job k,[4] that is, $L(D_k) = n$. However, if $Y_k \geq 1$, $L(D_k) \geq n+1$. Consequently, a down-crossing of level n can only occur at time D_k when $L(D_{k-1} = n+1)$ *and* $Y_k = 0$. It follows that the number of down-crossings up to time t is

$$D(n+1,0,t) = \sum_{k=1}^{D(t)} \mathbb{1}_{L(D_{k-1})=n+1} \mathbb{1}_{Y_k=0}. \tag{6.6.1}$$

FOR THE UP-CROSSINGS of level n, assume first that the system is in state m, $0 < m \leq n$, when the service of job k starts.[5] When $Y_k = 1$, it must be that $L(D_k) = m$ because job k left but one new job arrived in the meantime; thus, level n is *not* crossed. In fact, level n can only be up-crossed when $Y_k > n-m+1$. Thus,

$$D(m,n,t) = \sum_{k=1}^{D(t)} \mathbb{1}_{L(D_{k-1})=m} \mathbb{1}_{Y_k>n-m+1} \tag{6.6.2}$$

counts the number of up-crossings of level n for $m, 0 < m \leq n$.

When the system is in state $L(D_{k-1}) = 0$, there is a slight subtlety. We must first wait for job k to arrive[6] because job $k-1$ left an empty

[0] *So that the remaining service time is guaranteed to be* 0

[1] *Not $L(D_k-)$.*

[2] *Recall that level n lies between states n and $n+1$.*

[3] *Thus, $L(D_{k-1}) = n+1$.*

[4] *Namely, job k left.*

[5] *i.e., $L(D_{k-1}) = m$.*

[6] *But, as this is an arrival epoch, it is not captured by a change in the system state.*

system behind.[7] Once it arrived, $L(D_k) = 0$ when $Y_k = 0$, $L(D_k) = 1$ when $Y_k = 1$, and so on. Therefore,

$$D(0,n,t) = \sum_{k=1}^{D(t)} \mathbb{1}_{L(D_{k-1})=0} \mathbb{1}_{Y_k > n} \tag{6.6.3}$$

counts the number of up-crossings of that occur when $m = 0$.

[7] [6.6.1]–[6.6.2]

BY LEVEL-CROSSING we have that

$$D(n+1,0,t) = D(0,n,t) + \sum_{m=1}^{n} D(m,n,t) \pm 1 \text{ at most.} \tag{6.6.4}$$

Let us divide by t and take the limit $t \to \infty$. Using (5.2.3), we get

$$\frac{D(m,n,t)}{t} = \frac{D(t)}{t}\frac{D(m,t)}{D(t)}\frac{D(m,n,t)}{D(m,t)}, \quad 0 \le m \le n. \tag{6.6.5}$$

As before, $D(t)/t \to \delta$ and $D(m,t)/D(t) \to \delta(m)$. Then, by a reasoning similar to Section 6.5, $D(m,n,t)/D(m,t) \to G(n-m+1)$ for $0 < m \le n$, and $D(0,n,t)/D(0,t) \to G(n)$. Noting that $\pi(m) = \delta(m)$, see (5.3.9), and dividing by δ, we arrive at a recursion for $\{\pi(n)\}$:

$$f(0)\pi(n+1) = \pi(0)G(n) + \sum_{m=1}^{n} \pi(m)G(n+1-m). \tag{6.6.6}$$

Fig. 16 shows an example for level $n = 3$.

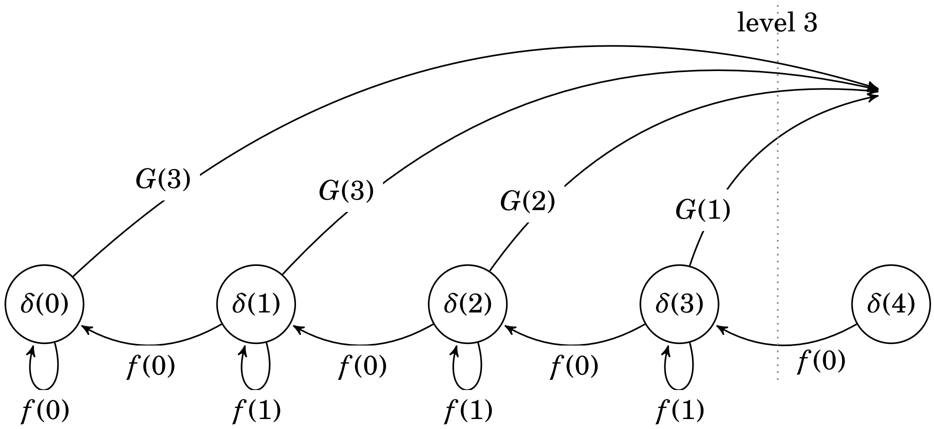


Figure 16: Level crossing at departure moments.

FOR THE EVALUATION of the above recursion we can just follow the scheme of Section 6.5, but there is an important difference, here the $\{f(k)\}$ needs to be computed.

We use a conditioning argument to find an expression for $\mathsf{P}\{Y = j\}$. Suppose we know that the service time is x. Then, $Y \sim P(\lambda x)$, hence,

$$\mathsf{P}\{Y_k = j \mid S = x\} = e^{-\lambda x}\frac{(\lambda x)^j}{j!}. \tag{6.6.7}$$

We write $\mathsf{P}\{S \in \mathrm{d}x\} = F(\mathrm{d}x) = F(x + \mathrm{d}x) - F(x)$ for the probability that the service time lies in the (infinitesimal) interval $[x, x + \mathrm{d}x]$.[8] When F

[8] *We shamelessly use infinitesimals here. We refer the interested student to any book on measure theory, to add a pile of technical details and hide the intuition.*

has a density f, then $F(\mathrm{d}x) = f(x)\,\mathrm{d}x$. With this,

$$\mathsf{P}\{Y_k = j\} = \int_0^\infty \mathsf{P}\{Y_k = j \,|\, S = x\} F(\mathrm{d}x) = \int_0^\infty e^{-\lambda x} \frac{(\lambda x)^j}{j!} F(\mathrm{d}x). \tag{6.6.8}$$

In simple cases we can carry out the integration by hand. A simple example is when $S \sim \mathrm{Exp}(\mu)$.[9] Another is to take $S \equiv s$, i.e., a constant. In this case, all probability mass is concentrated on s, so that $F(\mathrm{d}x) = 0$ for $x \neq s$ but $F(\mathrm{d}s) = \infty$.[10] With this, $f(j) = e^{-\lambda s}(\lambda s)^j/j!$.

[9] [6.6.3]

[10] *The density is the, so-called, δ function concentrated on s.*

When we cannot obtain a closed-form expression for the integral we need numerical methods. One simple method is as follows. Make a grid of size $\mathrm{d}x$, for some small number $\mathrm{d}x$, e.g. $\mathrm{d}x = 1/100$. Then take i such that $i\,\mathrm{d}x = x$, and write $\mathrm{d}F(i) = F(\mathrm{d}x) = F((i+1)\,\mathrm{d}x) - F(i\,\mathrm{d}x)$. With this,

$$\mathsf{P}\{Y_k = j\} \approx \sum_{i=0}^\infty e^{-\lambda i\,\mathrm{d}x} \frac{(\lambda i\,\mathrm{d}x)^j}{j!}\,\mathrm{d}F(i). \tag{6.6.9}$$

Let's try a numerical experiment.

```python
import numpy as np

labda = 3
mu = 4
j = 5
dx = 1 / 100


def F(x):
    return 1 - np.exp(-mu * x)


def dF(i):
    return F((i + 1) * dx) - F(i * dx)


def term(i):
    return (
        np.exp(-labda * i * dx)
        * (labda * i * dx) ** j
        / np.math.factorial(j)
        * dF(i)
    )


approx = sum(term(i) for i in range(50))
exact = mu / (labda + mu) * (labda / (labda + mu)) ** j
```

The value of the approximation is 0.0011159 while the exact value[11] is 0.0082619. The difference is significant. We can sum over more terms, for instance to 500. This gives 0.0080988. This is much better, but still

[11] [6.6.3]

the second digit is not correct, even though we evaluated 500 factorials, powers, and exponentials.[12]

To add precision we can make dx smaller and add yet more terms. However, it seems safer to use a real numerical integrator.

```
from scipy.integrate import quad


def g(x):
    return (
        np.exp(-labda * x)
        * (labda * x) ** j
        / np.math.factorial(j)
        * mu
        * np.exp(-mu * x)
    )


approx_2 = quad(g, 0, np.inf)
```

Now we obtain 0.0082619, which is indeed correct.

Ex 6.6.1. If $L(D_{k-1}) = 0$, what is $\mathsf{E}[D_k - D_{k-1}]$?

Ex 6.6.2. Show that if $LD_{k-1} = 0$ and $S_k \sim \text{Exp}(\mu)$, the density of $D_k - D_{k-1}$ is

$$f_{X+S}(t) = \frac{\lambda\mu}{\mu-\lambda}(e^{-\lambda t} - e^{-\mu t}). \tag{6.6.10}$$

Ex 6.6.3. If $S \sim \text{Exp}(\mu)$, show that

$$f(j) = \mathsf{P}\{Y_k = j\} = \frac{\mu}{\lambda+\mu}\left(\frac{\lambda}{\lambda+\mu}\right)^j. \tag{6.6.11}$$

Ex 6.6.4. If $S \sim \text{Exp}(\mu)$, show that

$$G(j) = \sum_{k=j+1}^{\infty} f(k) = \left(\frac{\lambda}{\lambda+\mu}\right)^{j+1}. \tag{6.6.12}$$

Ex 6.6.5. Check that the queue length distribution $\{\pi(n)\}$ of the $M/M/1$ queue satisfies (6.6.6).

This is a nice exercise to test your algebra skills.

12 *This must be numerically quite inefficient.*

7

QUEUEING CONTROL AND OPEN NETWORKS

In this chapter we study two topics: the control of an $M/M/1$ queue and $M/G/1$ by an N-policy in Sections 7.1 and 7.2, and open networks of $M/M/c$ stations in Section 7.3. As we will see, the analysis of the N-policy requires to solve an equation of the type $v = c + Pv$, where v and c are vectors and P a (stochastic) matrix, while for the networks we need to solve an equation of the type $\lambda = \gamma + \lambda P$, where λ and γ are (lying) vectors and P is again a (stochastic) matrix. Since these equations are just the transpose of each other, they allow us to study these two, seemingly unrelated, topics at once. We concentrate in Section 7.4 on the solution of these equations. The analysis in this chapter illustrates many tools and results of the previous chapters; as such, everything comes together here.

We point out that the techniques developed in this chapter extend (way) beyond just queueing theory; they are worth memorizing. The concepts we introduce here can for instance be generalized to (optimal) stopping problems, which find many applications beyond queueing, such as in finance, inventory theory, decision theory, and so on. As another set of extensions, it is possible to make the matrix P and the vector c depend on an action one can take in certain states. This idea underlies Markov decision theory, which in turn provides the theoretical basis of a number of machine learning tools such as Q learning, reinforcement learning, and so on. Thus, while this chapter closes our journey on the study of queueing systems, it is a first step toward a much longer journey on the diverse applications of probability theory.

7.1 N-POLICIES FOR THE $M/M/1$ QUEUE

In the queueing systems we analyzed up to now, the server is always present to start serving jobs at the moment they arrive. However, in cases in which there is a cost associated with changing the server from idle to busy, this condition is typically not satisfied. For instance, the cost to heat up an oven after being idle can be quite significant; in other cases, the operator of a machine has to move from one machine to another, which takes time. To reduce the average cost, an N-policy[0] can be used, works as follows. As soon as the system becomes empty,[1] the server switches off. Then it waits until N jobs have arrived,[2] and then it switches on. The server processes jobs until the system is empty again, switches off, and remains idle until N new jobs have arrived, and so on.[3]

Note that under an N-policy, when the load remains the same, the server has longer busy and idle times. In fact, some type of servers seem to use such policies on purpose. At hospitals, for instance, doctors prefer to let patients wait until the waiting room is quite full. Like this, the

[0] *[2.1.8]*

[1] *and the server idle*

[2] *or N or more items in case of batch arrivals*

[3] *Thus, an N-policy* controls *the server.*

doctor (server) does not have to wait for short times for patients that might be late, but instead can collect idle times into one long stretch, and do something (they find more) useful instead.

SUPPOSE IT COSTS K Euro to set up the server, independent of the time it has been idle, and each job charges h Euros per unit time while in the system. Then it makes sense to first build up a queue of N jobs right after the server becomes idle,[4] and after some time switch on the server to process jobs until the system is empty again.[5] The problem is to find a switching threshold N that minimizes long-run average cost.

[4] *To reduce time-average setup cost.*

[5] *To reduce time-average queueing costs of jobs.*

In this section we solve this problem for the $M/M/1$ queue, in the next section we generalize to the $M/G/1$ queue. In passing, we obtain a third way to compute the time-average number $\mathsf{E}[L]$ of jobs in the system; the first resulted from Little's law,[6] the second from the analysis of the $M/M/1$ queue in Section 6.1.

[6] *See [5.4.7]*

AS A FIRST STEP, we concentrate on the expected time $T(q)$ it takes to clear the system when there are q jobs present and the server switches on. In fact, we present three different ideas to obtain $T(q)$.

Firstly, we know that jobs arrive at rate λ and they are served at rate $\mu > \lambda$. Clearly, the net 'drain rate' of the queue is $\mu - \lambda$; hence we guess that $T(q) = q/(\mu - \lambda)$.[7] Observe that this reasoning ignores the stochasticity in arrival times and services. As such, it should not be applied in general; things can go very, very wrong.

[7] *By analogy: when you have a 'queue' of q km to cycle, and your speed is v km/h, then it takes q/v h to complete the trip.*

Secondly, when a job arrives to an empty system, it takes a busy time $\mathsf{E}[U]$ to get rid of this job and all jobs that arrive during the service. In other words, it takes $\mathsf{E}[U]$ time to move from 1 job in the system to 0, i.e., one less, jobs. But then, when there are q jobs in the system, it takes $T(q) = q\,\mathsf{E}[U]$ units of time to move to state 0. By [6.4.5], $\mathsf{E}[U] = \mathsf{E}[S]/(1-\rho) = 1/(\mu - \lambda)$, since $\mathsf{E}[S] = 1/\mu$ for the $M/M/1$ queue. Again, we see that $T(q) = q/(\mu - \lambda)$.

THE THIRD IDEA is the most powerful. Consider an arbitrary moment in time, $q > 0$, and the server is busy. Now either of two events happens first: a new job enters the system, or the job in service leaves. The probability of an arrival to occur first is $\alpha = \lambda/(\lambda + \mu)$,[8] the probability of a departure first is $\beta = 1 - \alpha = \mu/(\lambda + \mu)$. Moreover, the expected time to either an arrival or a departure, whichever is first, is $1/(\mu + \lambda)$.[9] Therefore, $T(q)$ must satisfy the following recursion:[10]

[8] *[2.4.14] and [6.6.3]*

[9] *[2.4.13]*

[10] *i.e., a difference equation.*

$$T(q) = \alpha T(q+1) + \beta T(q-1) + \frac{1}{\lambda + \mu}. \tag{7.1.1}$$

In words, the system stays in state q for an expected time $1/(\lambda + \mu)$ until an arrival or departure occurs. Then, it moves to state $q+1$ or $q-1$, and from there it takes $T(q+1)$ or $T(q-1)$ until the system is empty. Observe that this reasoning depends crucially on the memoryless property.

To solve this equation, we substitute the guess $T(q) = aq + b$ and solve for a and b. It is clear that $T(0) = 0$, hence $b = 0$.[11] It remains to solve for a. Filling in $T(q) = aq$ gives

[11] *If the system is empty, it takes no time to clear.*

$$aq = \alpha(aq + a) + \beta(aq - a) + 1/(\lambda + \mu). \tag{7.1.2}$$

Noticing that $\alpha+\beta=1$, this reduces to $0=a(\alpha-\beta)+1/(\lambda+\mu)$. Solving this for a gives right away that $a=1/(\mu-\lambda)$.

We note that the solution of (7.1.1) is unique once $T(0)$ is fixed. To see this, observe that in (7.1.1), $T(q+1)$ is a function of $T(q)$ and $T(q-1)$. Thus, $T(1)$ follows from $T(0)$, $T(2)$ from $T(1)$ and $T(0)$, and so on.

WITH THE SAME LINE of reasoning we can compute the expected cost $V(q)$, $q\geq 1$, to clear the system. Noting that the queueing cost is hq per unit time when there are q jobs in the system, it costs $hq/(\lambda+\mu)$ until an arrival or departure occurs. Hence, $V(q)$ satisfies the relation,

$$V(q)=\alpha V(q+1)+\beta V(q-1)+h\frac{q}{\lambda+\mu}. \tag{7.1.3}$$

We need a guess for $V(q)$ to solve this equation. Now observe that the last term in (7.1.1) is a constant, and that $T(q)$ is a linear function in q. As in (7.1.3) the last term is linear in q, let us try a quadratic in q for $V(q)$, i.e., $V(q)=aq^2+bq+c$. As $V(0)=0$, it follows already that $c=0$. Substituting $V(q)=aq^2+bq$ gives[12]

$$V(q)=\frac{h}{2}\frac{1}{\mu-\lambda}q^2+\frac{h}{2}\frac{\lambda+\mu}{(\mu-\lambda)^2}q. \tag{7.1.4}$$

AS AN IMMEDIATE application of the above, let us rederive $\mathsf{E}[L]=\rho/(1-\rho)$, i.e., (6.1.5). For this, we consider a *busy cycle* that results under the $N=1$ policy. Specifically, a cycle starts when a job arrives at an empty system. The server then switches on, and a busy period U starts. After some time,[13] the system becomes empty again, and the server idles for a period I. The cycle stops when a new job arrives. Write $C(1)$ for the expected duration of a busy cycle. Clearly, $C(1)=\mathsf{E}[I]+\mathsf{E}[U]=1/\lambda+T(1)$.[14]

With the renewal-reward theorem it is simple to see that[15]

$$\frac{V(1)}{C(1)}=\frac{V(1)}{1/\lambda+T(1)}=h\,\mathsf{E}[L]. \tag{7.1.5}$$

After some algebra we get that $V(1)/C(1)=h\rho/(1-\rho)$, thereby completing the argument.[16]

LET US NEXT analyze the cost under a general N-policy. As we already have expressions for the time and cost while the server is on, we only have to consider the time and cost while the server is off.

Clearly, right after the server switches off, we need N independent inter-arrival times to reach level N, which takes N/λ units of time in expectation.[17].

For the cost during the build up the queue, we use again a recursive procedure. Write $W(q)$ for the accumulated queueing cost[18] from the moment the server becomes idle up to the arrival time of the qth job (the job that sees $q-1$ jobs in the system). Then,[19]

$$W(q)=W(q-1)+h\frac{q-1}{\lambda}=h\frac{q(q-1)}{2\lambda}. \tag{7.1.6}$$

12 *[7.1.1]*

13 *In expectation* $\mathsf{E}[U]=T(1)$.

14 *[6.4.5]*

15 *[7.1.2]*

16 *[7.1.3]*

17 *[2.4.9]*

18 *Here W is not the waiting time in queue.*

19 *[7.1.4]*

It remains to assemble all results. Let us assume that the switching cost is K. Then, by the renewal-reward theorem, the time-average cost of the N-policy is equal to

$$\frac{W(N)+K+V(N)}{N/\lambda+T(N)}, \tag{7.1.7}$$

since the expected cycle duration is $C(N)=N/\lambda+T(N)$.

FINDING THE OPTIMAL N is easy. Observe that $V(N)$ and $W(N)$ are quadratic in N, while $C(N)$ is linear in N. Hence, the average cost is a convex function of N. In Section 7.2 we derive the general expression and identify the optimal level N^*.

Ex 7.1.1. Derive the expression for $V(q)$.

Ex 7.1.2. Explain that $V(1)/C(1)=h\,\mathsf{E}[L]$.

Ex 7.1.3. Show that $V(1)/C(1)=h\rho/(1-\rho)$.

Ex 7.1.4. Explain the recursion for $W(q)$ and solve it.

7.2 N-POLICIES FOR THE $M/G/1$ QUEUE

Interestingly, we can extend the analysis of Section 6.6 and Section 7.1 to compute the average cost of the $M/G/1$ queue under an N-policy. At the end we will find the threshold N^* that minimizes the long-run average cost under the N-policy.

Throughout we consider the $M/G/1$ queueing process at moments at which services start. This is similar to Section 6.6: When the system is not empty, these are departure epochs.

IT IS EASY to obtain an expression for the clearing time $T(q)$ when a job starts with q jobs in the system. As in Section 6.6, write Y for the number of jobs that arrive during a service time. Then, analogous to (7.1.1), $T(q)$ satisfies the relation

$$T(q)=\mathsf{E}[S]+\mathsf{E}[T(q+Y-1)], \tag{7.2.1}$$

because first the job in service must leave, and then, when $Y=k$, it takes $T(q+k-1)$ to hit level 0.

Again, we guess that $T(q)=aq+b$. But $b=0$ since $T(0)=0$. Substitution of aq gives $0=a\,\mathsf{E}[Y]-a+\mathsf{E}[S]$ and noting that $\mathsf{E}[Y]=\lambda\,\mathsf{E}[S]$,[0] we directly obtain

[0] [4.4.5]

$$T(q)=\frac{\mathsf{E}[S]}{1-\lambda\,\mathsf{E}[S]}q, \tag{7.2.2}$$

TO FIND THE cost to clear the queue, we first concentrate on the expected queueing cost $H(q)$ that accrue during the service time of the job. This consists of two components. The first is the expected cost of the q jobs in the system while the job is in service. Clearly, this is $hq\,\mathsf{E}[S]$. The second component is the cost of new jobs that arrive during the service. While this is slightly harder to determine, we can combine the ideas underlying the derivation of (7.1.6) with (2.2.4).

Suppose that, when a job arrives during a service with remaining time s, it has to pay hs directly upon arrival. Let $U(s)$ denote the expected amount paid by all job that arrive during a service time s. If during an interval of length δ, $0 < \delta \ll 1$, no new jobs arrive, the expected amount paid is $U(s-\delta)$.[1] If one job arrives, the total amount must be $U(s-\delta)+\lambda s$, and so on. With probability $1-\lambda\delta$ no job arrives, with probability $\lambda\delta$ one job arrives, and since $\delta \ll 1$, we neglect the probability of multiple arrivals. Therefore,

$$U(s) = U(s-\delta) + (1-\lambda\delta)\cdot 0 + \lambda\delta h s + o(\delta). \tag{7.2.3}$$

By subtracting $U(s-\delta)$ from both sides, dividing by δ, and taking the limit $\delta \to 0$, this becomes the differential equation $U'(s) = \lambda h s$.[2] As $U(0) = 0$, we get that $U(s) = \lambda h s^2/2$.

Since $U(s)$ is the expected total cost given that the service time is s, $\mathsf{E}[U(S)] = \lambda h \mathsf{E}\left[S^2\right]/2$ is the expected queueing cost of the arrivals during a service time.

By combining the first and second component of $H(q)$, we obtain

$$H(q) = hq\,\mathsf{E}[S] + \frac{1}{2}\lambda h\,\mathsf{E}\left[S^2\right]. \tag{7.2.4}$$

LET $V(q)$ BE the expected queueing costs until the system is empty and the service starts with q in the system. By analogy with (7.2.1), $V(q)$ must be the solution of

$$V(q) = \mathsf{E}[V(q+Y-1)] + H(q). \tag{7.2.5}$$

Now note that as in (7.1.3), the cost $H(q)$ has a term linear in q and a constant term. As before, we substitute the form $V(q) = aq^2 + bq + c$, assemble terms with the same power in q, and solve for the coefficients. After some work we arrive at[3]

$$V(q) = \frac{h}{2}\frac{\mathsf{E}[S]}{1-\rho}q^2 + h\frac{1+\rho C_s^2}{2}\frac{\mathsf{E}[S]}{(1-\rho)^2}q. \tag{7.2.6}$$

NOW THAT WE have the expected clearing time and cost, we can compute the long-run average cost under a general N-policy. The cycle time $C(N) = N/\lambda + T(N)$, and the queueing cost $W(N)$ until level N is reached while the server is idle is given by (7.1.6). Combining all this results in the long-run average cost[4]

$$\frac{W(N)+K+V(N)}{C(N)} = h\frac{1+C_s^2}{2}\frac{\rho^2}{1-\rho} + h\rho + h\frac{N-1}{2} + K\frac{\lambda(1-\rho)}{N}. \tag{7.2.7}$$

From this, the PK-formula follows directly.[5]

Finally, minimizing over N gives that

$$N^* \approx \sqrt{\frac{2\lambda(1-\rho)K}{h}}. \tag{7.2.8}$$

The expression for N^* is a famous result in inventory theory: it is the optimal order size for a machine that can switch on at cost K and an item has holding cost h per unit time. The demand is λ and the production time of an item is $\mathsf{E}[S]$. N^* is known as the *Economic Production Quantity (EPQ)*. Taking $\mathsf{E}[S] \to 0$, hence $\rho \to 0$, N^* reduces to the *Economic Order Quantity (EPQ)* $\sqrt{2\lambda K/h}$.

[1] *Because jobs pay at the moment they arrive.*

[2] *Assuming that $U(\cdot)$ is differentiable.*

[3] [7.2.3]–[7.2.4]

[4] [7.2.7]

[5] [7.2.6]

Ex 7.2.1. Explain intuitively that the system rate-stable for any N.

Ex 7.2.2. Why does the utilization ρ not depend on N?

Ex 7.2.3. Simplify $aq^2 + bq = a\,\mathsf{E}\left[(q+Y-1)^2\right] + b\,\mathsf{E}[q+Y-1] + H(q)$, and assemble powers in q to obtain:

$$a = \frac{h}{2}\frac{\mathsf{E}[S]}{1-\mathsf{E}[Y]} = \frac{h}{2}\frac{\mathsf{E}[S]}{1-\rho}, \tag{7.2.9}$$

$$b(1-\mathsf{E}[Y]) = a(\mathsf{E}\left[Y^2\right] - 2\,\mathsf{E}[Y] + 1) + \frac{1}{2}h\lambda\,\mathsf{E}\left[S^2\right]. \tag{7.2.10}$$

Ex 7.2.4. Derive the expression for $V(q)$ with the previous exercise.

In a sense, this is trivial, as it is just algebra, but it is hard to get the details right.

Ex 7.2.5. Check that $V(q)$ reduces to that of the $M/M/1$ queue.

Ex 7.2.6. Derive the PK formula from (7.2.7).

Ex 7.2.7. Derive (7.2.7).

7.3 OPEN SINGLE-CLASS PRODUCT-FORM NETWORKS

Up to now our analysis focused on single-station queueing systems. However, jobs, or patients, may need several 'processing' steps at different stations. In such cases, jobs move from one queueing system to another. In this section we analyze such queueing networks in simple settings. We start with two stations in tandem, and then extend to general networks. Throughout we assume that external jobs[0] arrive as a Poisson processes, and that service times at all stations are exponentially distributed. Under this condition we are able to obtain closed-form expressions for the stationary distribution of jobs at each station.[1]

[0] *i.e., new jobs that have not visited any other station before.*

[1] *Recall, Section 4.5 only considers the expected sojourn times in tandem networks of G/G/c queues, not the distribution of the number of jobs at each station in a network of M/M/1 queues.*

WE START WITH stating the remarkable, and crucially important, result that the *inter-departure times* of jobs of an $M/M/1$ queue are exponentially distributed with rate λ, just as the inter-arrival times.[2] This property makes analysis of a tandem network[3] of stations, i.e., stations placed in sequence, particularly easy.

[2] *[7.3.1]*

[3] *Tandem network*

When the first station is an $M/M/1$ queue, jobs arrive as a Poisson process, but its output process is also a Poisson process. Therefore the second station, i.e., the station immediately downstream of station 1, receives jobs as a Poisson process. If job service times at the second station are exponentially distributed, then this station is also an $M/M/1$ station. But then its departure process is a Poisson process in turn, and the third station receives jobs as a Poisson process, and so on.

With this insight, it follows from (6.1.5) that the expected total sojourn time in a tandem network of M stations is equal to

$$\mathsf{E}[J] = \sum_{i=1}^{M} \mathsf{E}[J_i] = \sum_{i=1}^{M} \frac{\mathsf{E}[S_i]}{1-\rho_i}, \tag{7.3.1}$$

where $\mathsf{E}[J_i]$ is the sojourn time at station i with $\rho_i = \lambda\,\mathsf{E}[S_i]$.

IT IS NOT DIFFICULT to extend the above result to general networks of $M/M/1$ queues. For this, we first need to model such networks more formally. In particular, we assume that the probability that a job moves to station j after completing its service at station i is independent of anything else, and is given by the number $P_{ij} \in [0,1]$.[4] We assemble all these probabilities in a *routing matrix* P such that P_{ij} is the element of P on the ith row and jth column. Define $P_{i0} = 1 - \sum_j P_{ij}$ as the probability that a job leaves the network from station i. For the network to be stable, we need to require, see Section 7.4,

$$P_{i0} \in [0,1] \text{ for all } i, \text{ and } P_{i0} > 0 \text{ for at least one } i. \tag{7.3.2}$$

[4] *This is called Markov routing.*

Consider station i, say, and assume that jobs arrive as a Poisson process with rate λ_i. Since service times are exponentially distributed, the departure process is also Poisson with rate λ_i. Then, after departure, jobs are sent with probability P_{ij} to station j. It follows from [2.2.17] that jobs sent to station j from station i form a thinned Poisson process with rate $\lambda_i P_{ij}$.

Now take the perspective of station j. This station receives a merged 'stream' of thinned Poisson process from all other stations. But, by [2.2.12], this merged process is also a Poisson process with rate $\sum_{i=1}^M \lambda_i P_{ij}$. Above we assume that external jobs arrive at station j as a Poisson process with rate γ_j. Merging this process with the other Poisson stream results in the Poisson process with rate $\gamma_j + \sum_{i=1}^M \lambda_i P_{ij}$.

By assumption, service times at station j are exponential, hence, the departure process of this station is Poisson. But then the thinned process resulting from routing its departing jobs to yet other stations is again Poisson, and so on.

We arrive at the intuitively clear fact that this network behaves as a set of $M/M/1$ queues. Below we will give a formal proof of this fact for two stations.

Note that we allow for external jobs arriving at any station. Therefore, this is an open network[5]. This differs from so-called *closed networks*; in such networks jobs do not enter or leave the network.

[5] *Open network*

IT IS EVIDENT that, when the network is stable, all jobs that enter a station eventually must leave that station. This insight leads us to the traffic equations[6], which state that for each station i the departure rate must equal the arrival rate, i.e.,

[6] *Traffic equations*

$$\lambda_i = \gamma_i + \sum_{j=1}^M \lambda_j P_{ji}, \quad i = 1, \ldots, M. \tag{7.3.3}$$

For the *network as whole*, the total external arrival rate is given by $|\gamma| = \sum_{i=1}^M \gamma_i$. Hence, this is also the departure rate of the network.[7]

[7] *Not for an individual station i, because $\delta_i = \lambda_i$.*

LET US FOR the moment assume that we can solve the traffic equations, in other words, for given $\gamma = (\gamma_1, \ldots, \gamma_M)$ and routing matrix P we can find a set of numbers $\lambda = (\lambda_1, \ldots, \lambda_M)$ that satisfies (7.3.3). Then, we can define $\rho_i = \lambda_i \mathsf{E}[S_i]$ as the utilization of (the server of) station i. Clearly, we assume that $\rho_i < 1$ for all stations i.

Evidently, the average total number of jobs in this network of $M/M/1$ stations is equal to the sum of the average number of jobs at each station, hence, $\mathsf{E}[L] = \sum_{i=1}^{M} \mathsf{E}[L_i]$. With this we can also find an expression for the expected total sojourn time in the network $\mathsf{E}[J]$. Applying Little's law first to the network as a whole and then to each station individually, we see that

$$|\gamma|\,\mathsf{E}[J] = \mathsf{E}[L] = \sum_{i=1}^{M} \mathsf{E}[L_i] = \sum_{i=1}^{M} \lambda_i\,\mathsf{E}[J_i] = \sum_{i=1}^{M} \lambda_i \frac{\mathsf{E}[S_i]}{1-\rho_i}. \tag{7.3.4}$$

Dividing by $|\gamma|$ gives $\mathsf{E}[J]$ in terms of the visit ratios[8] $\lambda_i/|\gamma|$,

$$\mathsf{E}[J] = \sum_{i=1}^{M} \frac{\lambda_i}{|\gamma|} \frac{\mathsf{E}[S_i]}{1-\rho_i}. \tag{7.3.5}$$

[8] *Visit ratios*

This is intuitive: the visit ratio $\lambda_i/|\gamma|$ tells how often station i is visited relative to the total number of arrivals. Thus, $\mathsf{E}[J_i]\lambda_i/|\gamma|$ is the amount of time an 'average' job spends at station i before leaving the network.

ABOVE WE DERIVED expressions for the average waiting time in a network of $M/M/1$ queues, but it is possible to obtain a much stronger result. In fact, the stationary probability $p(n)$ that the system contains $n = (n_1, n_2, \ldots, n_M)$ at stations $1, \ldots, M$ can be written as the *product* of the probabilities of the individual stations, specifically,

$$p(n) = \mathsf{P}\{N_1 = n_1, \ldots, N_M = n_M\} = \Pi_{i=1}^{M} p(n_i). \tag{7.3.6}$$

where $p(n_i) = (1-\rho_i)\rho_i^{n_i}$ is the stationary probability that station i contains n_i jobs, compare (6.1.4). In other words, the stationary probability $p(n_i)$ of station i is *independent* of the state of the other stations. Hence, in stationarity, we can concentrate on each station individually. As a consequence, we can easily compute the excess probability $\mathsf{P}\{L_1 > n_1\}$ for each station i.

As a matter of fact, any stable open network of $M/M/c$ stations[9] admits a product-form solution[10], a result known as Jackson's theorem[11].

[9] *Note, $M/M/c$ queues, not just $M/M/1$ queues.*

[10] *Product-form solution*

[11] *Jackson's theorem*

LET US PROVE this result for the case of two $M/M/1$ stations in tandem. Let $p(i,j) = \mathsf{P}\{N_1 = i, N_2 = j\}$ be the probability that station 1 contains i jobs and station 2 contains j jobs. Specifically, we have to show that when $p(i,j)$ has the form $(1-\rho_1)(1-\rho_2)\rho_1^i\rho_2^j$, the balance equations are satisfied for all $i, j \geq 0$, see Section 5.2.

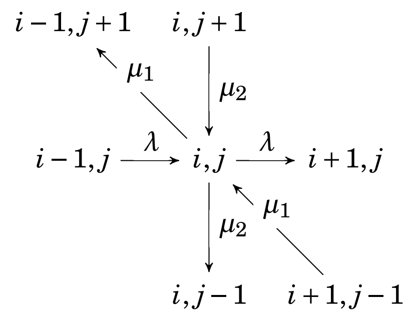


Figure 17: Transitions in two stations in tandem.

Here we focus on the case $i, j \geq 1$, see Fig. 17. As the normalization factor appears at both sides of the balance equations, we write $p(i,j) = \rho_1^i\rho_2^j$ for ease. Then it follows that, Clearly, the balance equations hold for $i, j \geq 1$.

$$\begin{aligned}
\text{rate out } &= (\lambda + \mu_1 + \mu_2)p(i,j) && (7.3.7)\\
&= \lambda\rho_1^i\rho_2^j + \mu_1\rho_1^i\rho_2^j + \mu_2\rho_1^i\rho_2^j && (7.3.8)\\
&= \mu_2\rho_1^i\rho_2^{j+1} + \lambda\rho_1^{i-1}\rho_2^j + \mu_1\rho_1^{i+1}\rho_2^{j-1} && (7.3.9)\\
&= \mu_2 p(i,j+1) + \lambda p(i-1,j) + \mu_1 p(i+1,j-1) && (7.3.10)\\
&= \text{ rate in.} && (7.3.11)
\end{aligned}$$

It remains to check the boundary[12] to complete the claim.

[12] *[7.3.4]–[7.3.6]*

Ex 7.3.1. Show that the inter-departure times $D_k - D_{k-1} \sim \text{Exp}(\lambda)$.

Ex 7.3.2. Why do we require (7.3.2)?

Ex 7.3.3. We have a two-station single-server open network. Jobs enter the network at the first station with rate γ. A fraction α returns from station 1 to itself; the rest moves to station 2. At station 2 a fraction β_2 returns to station 2 again, a fraction β_1 goes to station 1.

First, compute λ, then analyze what happens if $\alpha \to 1$ or $\beta_1 \to 0$.

Ex 7.3.4. Check that $p(0,0)$ satisfy the balance equation for state $(0,0)$.

Ex 7.3.5. Check that $p(i,0)$ satisfy the balance equation for state $(i,0)$.

Ex 7.3.6. Check that $p(0,j)$ satisfy the balance equation for state $(0,j)$.

7.4 ON $\lambda = \gamma + \lambda P$

There is a remarkable similarity between (7.1.3) and (7.3.3). To see this, write the former in matrix form as $v = Pv + h$, with

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots \\ \beta & 0 & \alpha & 0 & \dots \\ 0 & \beta & 0 & \alpha & 0 \\ \vdots & & \ddots & \ddots & \ddots \end{pmatrix}, \qquad h = \begin{pmatrix} 0 \\ 1/(\lambda+\mu) \\ 2/(\lambda+\mu) \\ \vdots \end{pmatrix}, \tag{7.4.1}$$

and the latter as $\lambda = \gamma + \lambda P$.[0] Clearly, the former is just the transpose of the latter. In this final section we concentrate on finding a general condition to ensure that these equations have a solution.

[0] *A similar form can be found for (7.2.5).*

We remark that, formally speaking, the matrix P in $v = Pv + h$ is an infinite matrix. Rather than dealing with the ensuing technical complications, we bypass them by restricting the analysis to an $M/M/1/K$ queue under an N-policy, with $K \gg N$ for any N that can be 'reasonable'.[1]

[1] *What is reasonable depends on the context of course, but $N = 1000$ seems ridiculously large for any practical queueing system.*

For ease of writing, we only regard $\lambda = \gamma + \lambda P$.

TO START, define iteratively, $\lambda^0 = 0$, $\lambda^n = \gamma + \lambda^{n-1} P$, for $n \geq 1$. Then, by substitution, we find

$$\lambda^n = \gamma + \lambda^{n-1} P = \gamma + (\gamma + \lambda^{n-2} P) P = \dots = \gamma \sum_{m=0}^{n-1} P^m, \tag{7.4.2}$$

where P^0 as the identity matrix. Since P is a non-negative matrix, P^n is also non-negative for any n, hence, $\sum_{m=0}^{n} P^m$ is a monotone increasing sequence (of matrices).[2] Thus, for $\lambda = \gamma + \lambda P$ to have a (unique) solution, this sum must remain finite as $n \to \infty$.

[2] *Formally, non-decreasing sequence.*

Now, observe the similarity with $\sum_{i=0}^{n} x^i$ with $|x| < 1$. From (1.2.3d) we see that this converges to $1/(1-x)$ for $n \to \infty$. By the same token, if there is an $\epsilon > 0$ such that for all i,j, $0 \leq P_{ij}^m \leq (1-\epsilon)^m$, then $0 \leq \sum_{m=0}^{n} P_{ij}^m \leq \sum_{m=0}^{n} (1-\epsilon)^m \leq 1/\epsilon$.

It remains to find a condition to ensure that such ϵ exists. For this, we discuss two ways.

FROM [7.3.2] WE KNOW that $P_{i0} = 1 - \sum_{j=1}^{M} P_{ij}$ is the probability that a job leaves the network from station i. Next, consider a station k with $P_{ki} > 0$. Then the probability that a job starts at k and leaves the network after a visit to i is $P_{ki}P_{i0} > 0$. Consequently, $P^2_{k0} = \sum_{j=0}^{M} P_{kj}P_{j0} \geq P_{ki}P_{i0} > 0$.

More generally, we say that $M \times M$ matrix P is transient[3] when it is possible to leave the network from any station in at most M steps. In other words, for any station j there is a sequence of intermediate stations $j_1, j_2, \ldots, j_{M-1}$ such that $P^M_{j0} \geq P_{jj_1}P_{j_1j_2}\cdots P_{j_{M-1}0} > 0$. Using this, it is not hard to prove that $P^n \leq (1-\epsilon)^n$ when P is transient.[4]

[3] *Transient*

[4] *[7.4.2]*

FOR THE SECOND method, let us make the simplifying assumption that P is a diagonalizable matrix with M different eigenvalues.[5] In this case, there exists an invertible matrix V with the (left) eigenvectors of P as its rows and a diagonal matrix Λ with the eigenvalues of P on its diagonal such that $VP = \Lambda V$. Hence, premultiplying with V^{-1}, $P = V^{-1}\Lambda V$. But then $P^2 = V^{-1}\Lambda V \cdot V^{-1}\Lambda V = V^{-1}\Lambda^2 V$, and in general $P^n = V^{-1}\Lambda^n V$. Clearly, if each eigenvalue λ_i is such that its modulus $|\lambda_i| < 1$, then $\Lambda^n \to 0$ exponentially fast, hence $P^n \to 0$ exponentially fast.

[5] *The argument below applies just as well matrices reduced to Jordan normal form, but adds only notational clutter.*

So, let us prove that all eigenvalues of a finite, transient routing matrix P have modulus less than 1. For this we use Gerschgorin's disk theorem[6]. Define the Gerschgorin disk of the ith row of the matrix P as the disk in the complex plan:

[6] *Gerschgorin's disk theorem*

$$B_i = \left\{ z \in \mathbb{C}; |z - p_{ii}| \leq \sum_{j \neq i} |p_{ij}| \right\}. \tag{7.4.3}$$

In words, this is the set of complex numbers that lies within a distance $\sum_{j\neq i} |p_{ij}|$ of the point p_{ii}. Next, assume for notational simplicity that for each row i of P we have that $\sum_j p_{ij} < 1$.[7] Then this implies for all i that

[7] *Otherwise apply the argument to P^M.*

$$1 > \sum_{j=1}^{M} p_{ij} = p_{ii} + \sum_{j \neq i} p_{ij}. \tag{7.4.4}$$

Since all elements of P are non-negative, so that $|p_{ij}| = p_{ij}$, it follows that

$$-1 < p_{ii} - \sum_{j\neq i} p_{ij} \leq p_{ii} + \sum_{j \neq i} p_{ij} < 1. \tag{7.4.5}$$

With this and using that p_{ii} is a real number (so that it lies on the real number axis) it follows that the disk B_i lies strictly within the complex unit circle $\{z \in \mathbb{C}; |z| \leq 1\}$. As this applies to any row i, the union of the disks $\cup_i B_i$ also lies strictly within the complex unit circle. Now we invoke Gerschgorin's theorem, which states that all eigenvalues of the matrix P must lie in $\cup_i B_i$, to conclude that all eigenvalues of P lie strictly in the unit circle, hence all eigenvalues have modulus smaller than 1.

HERE WE FINISH our discussion of queueing systems, but there are many other interesting extensions to learn, for which we refer to the following work.

- You can find really nice discussions of networks of $M/M/\infty$, chemical reactions, population dynamics and Petri nets in Baez and Biamonte [2019], which is freely available on arXiv.

- Simple queueing networks (networks that satisfy so-called local balance) can be modeled as electrical networks. For this, see Doyle and Laurie Snell [1984], which you can download for free from the homepage of Doyle.

- In more general terms, queueing systems or networks are examples of Markov processes. A particularly nice book on these topics is Norris [1997]. The material of this chapter can be couched in the theory of martingales and optimal stopping. Besides that this is nice theory, this is widely used in quantitative finance.

Ex 7.4.1. What is the routing matrix P for a tandem network with M stations? Show that $P^M = 0$.

Ex 7.4.2. Show that when P is transient, $P^n \leq (1-\epsilon)^n$ for some $\epsilon > 0$.

Ex 7.4.3. Why does the assumption of [7.4.2] does not apply to an infinite transient matrix P?

Ex 7.4.4. Show that the matrix P used in Section 7.1 is transient for a finite system.

h.1.2.6. Write $\sum_{k=0}^{\infty} G(k) = \sum_{k=0}^{\infty} \sum_{m=k+1}^{\infty} \mathsf{P}\{X = m\}$, reverse the summations. Then realize that $\sum_{k=0}^{\infty} \mathbb{1}_{k<m} = m$.

h.1.2.7. $\sum_{i=0}^{\infty} iG(i) = \sum_{n=0}^{\infty} \mathsf{P}\{X = n\} \sum_{i=0}^{\infty} i\, \mathbb{1}_{n \geq i+1}$, and reverse the summations.

h.1.2.8. $\mathsf{E}[X] = \int_0^{\infty} x F(\mathrm{d}x) = \int_0^{\infty} \int_0^{\infty} \mathbb{1}_{y \leq x} \,\mathrm{d}y F(\mathrm{d}x)$.

h.1.2.9. $\int_0^{\infty} y G(y) \,\mathrm{d}y = \int_0^{\infty} y \int_0^{\infty} \mathbb{1}_{y \leq x} f(x) \,\mathrm{d}x \,\mathrm{d}y$.

h.1.2.12. For the second child, condition on the event that the first does not choose the right number. Use the definition of conditional probability: $\mathsf{P}\{A \,|\, B\} = \mathsf{P}\{AB\}/\mathsf{P}\{B\}$ provided $\mathsf{P}\{B\} > 0$.

h.2.1.2. Modify d_k in (2.1.1) to incorporate the changed service behavior. Then, substitute d_k in the expression for L_k.

h.2.1.5. Make a queue for the new and repaired items and use [2.1.10].

h.2.1.6. Use [2.1.2] for the dynamics of $\{L_k\}$.

h.2.1.8. Introduce a variable $I_k \in \{0, 1\}$ to keep track of the state of the server. Then, $I_{k+1} = \mathbb{1}_{L_k \geq N} + I_k \mathbb{1}_{0 < L_k < N}$ implements the N-policy. Now use [2.1.7] to find $\{c_k\}$ and the switching cost.

h.2.1.9. Consider a numerical example. Suppose $L_{k-1} = 20$. Suppose that the capacity is $c_k = 3$ for all k. Then a job that arrives in the kth period, must wait at least 20/3 (plus rounding) periods before it can leave the system. Now generalize this numerical example.

h.2.1.10. Compute the number of jobs that depart from queue 1. Subtract the used capacity for these jobs from the total capacity to get the capacity remaining for queue 2.

h.2.1.18. Introduce an extra variable p_k that specifies which queue is being served. If $p_k = 0$, the server is moving from one queue to the other, if $p_k = 1$, the server is at queue 1, and if $p_k = 2$ it is at queue 2. Develop a table with conditions on L_k^i and p_k to specify p_{k+1}.

h.2.2.1. Use that $\mathsf{E}[X + Y] = \mathsf{E}[X] + \mathsf{E}[Y]$.

h.2.2.2. First find p, n, λ and t such that the rate at which an event occurs in both processes are the same. Then consider the binomial distribution and use the standard limit $(1 - x/n)^n \to e^{-x}$ as $n \to \infty$.

h.2.2.4. Use the definition of the conditional probability and small o notation.

h.2.2.5. Use [2.2.4] and [1.2.1].

h.2.2.6. See the hint for [2.2.5], and use $\sum_{i=2}^{\infty} x^i/i! = \sum_{i=0}^{\infty} x^i/i! - x - 1 = e^x - x - 1$.

h.2.2.7. Use (1.2.4d). Note that the term with $n = 0$ does not contribute in the following summation

$$\sum_{n=0}^{\infty} n\frac{\lambda^n}{n!} = \sum_{n=1}^{\infty} n\frac{\lambda^n}{n!} = \sum_{n=1}^{\infty} \frac{\lambda^n}{(n-1)!} = \lambda \sum_{n=0}^{\infty} \frac{\lambda^n}{n!} = \lambda e^{\lambda}, \tag{7.4.6}$$

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

h.2.2.9. Use (1.2.4e), [2.2.8], and [2.2.7].

h.2.2.10. Observe that

$$\mathbb{1}_{N(0,s]+N(s,t]=1}\,\mathbb{1}_{N(0,s]=1} = \mathbb{1}_{1+N(s,t]=1}\,\mathbb{1}_{N(0,s]=1} = \mathbb{1}_{N(s,t]=0}\,\mathbb{1}_{N(0,s]=1}. \tag{7.4.7}$$

Use independence and (1.2.4b).

h.2.2.12. Use sets $\{N_\lambda(t) = i\}$ to decompose $\{N_\lambda(t) + N_\mu(t) = n\}$. With this observe that

$$\mathbb{1}_{N_\lambda(t)+N_\mu(t)=n} = \sum_{i=0}^{n} \mathbb{1}_{N_\lambda(t)=i, N_\mu(t)=n-i}. \tag{7.4.8}$$

Take expectations left and right, use (1.2.4b), and independence of N_λ and N_μ. Near the end of the computation, use (1.2.3a).

h.2.2.13. Use the standard formula for conditional probability and that $N_\lambda(t) + N_\mu(t) \sim \mathsf{P}((\lambda+\mu)t)$. Interpret the result.

h.2.2.14. Suppose that N_1 is the thinned stream, and N the original stream. Condition on the total number of arrivals $N(t) = n$ up to time t. Then, realize that the probability that a person is of type 1 is p. Hence, when you consider n people in total, the number $N_1(t)$ of type 1 people is binomially distributed. Thus, given that n people arrived, the probability of k 'successes' (i.e., arrivals of type 1), is

$$\mathsf{P}\{N_1(t) = k \,|\, N(t) = n\} = \binom{n}{k} p^k (1-p)^{n-k}. \tag{7.4.9}$$

Use (1.2.8) to decompose the $\{N_1 = k\}$, and (1.2.3c) at the end.

h.2.2.15. Use (1.2.4d) with $f(x) = e^{sx}$.

h.2.2.16. Use (1.2.7b) and (1.2.7c).

h.2.2.17. Dropping the dependence of N on t for the moment for notational convenience, consider the random variable

$$Y = \sum_{i=1}^{N} Z_i, \tag{7.4.10}$$

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.

h.2.2.18. Use (1.2.7d) and (1.2.7a).

h.2.2.19. Solve [2.2.15] and [2.2.17] first. Perhaps [2.2.2] is also useful.

h.2.3.1. What is $\mathbb{1}_{A_k \leq t}$ if $A_k \leq t$?

h.2.3.4. Compare this to the definition in (2.3.7).

h.2.3.5. 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.2.3.6. Observe that jobs arrive faster than they are served.

h.2.3.7. Use (2.3.3).

h.2.3.9. 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.2.3.13. Use Boolean algebra. Write, for notational ease, $A = \mathbb{1}_{A_k \leq t}$ and $\bar{A} = 1 - A = \mathbb{1}_{A_k > t}$, and define something similar for D. Then show that $A - D = A\bar{D} - \bar{A}D$, and show that $\bar{A}D = 0$. Finally sum over k.

h.2.3.14. Use that $L(A_k) > 0$ means that the system contains at least one job at the time of the kth arrival, and that $A_k \leq D_{k-1}$ means that job k arrives before job $k-1$ departs.

h.2.3.17. 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.

h.2.4.1.

$$\mathsf{E}[X] = \int_0^\infty t f(t)\, \mathrm{d}t = \int_0^\infty t\lambda e^{-\lambda t}\, \mathrm{d}t, \tag{7.4.11}$$

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

h.2.4.2. Use [2.4.1].

h.2.4.3. Use [2.4.1] and [2.4.2].

h.2.4.5. Use (1.2.7b) and (1.2.7c).

h.2.4.6. Use the definition (2.2.5)

h.2.4.7. Simplify $\mathsf{P}\{X > t+h \,|\, X > t\}$ with $\mathsf{P}\{A \,|\, B\} = \mathsf{P}\{AB\}/\mathsf{P}\{B\}$.

h.2.4.8. $N(t) = 0 \iff X_1 > t$.

h.2.4.9. $\mathsf{E}[A_i] = \mathsf{E}\left[\sum_{k=1}^i X_k\right]$

h.2.4.10. Why is $M_{A_i}(t) = \mathsf{E}\left[e^{tA_i}\right] = \prod_{k=1}^i \mathsf{E}\left[e^{tX_k}\right]$?

h.2.4.11. Use [2.4.1], or $\mathsf{E}[X] = \frac{\mathrm{d}}{\mathrm{d}t} M_X(t)|_{t=0}$.

h.2.4.12. $\mathsf{P}\{N(t)=k\} = \mathsf{P}\{A_k \leq t\} - \mathsf{P}\{A_{k+1} \leq t\}$.

h.2.4.13. $\min\{X,S\} > x \iff X > x \,\&\, S > x$.

h.2.4.14. Use the joint distribution of X and S, or use [2.2.13].

h.3.2.1. Use recursion and use subsequently, $\max\{\max\{a,b\},c\} = \max\{a,b,c\}$, $L_0 = Z_0$, $\max\{-a,-b\} = -\min\{a,b\}$.

h.3.2.3. Let $W_n = \sum_{k=1}^n S_k$. Since $\{S_k\}$ are assumed to be i.i.d. for the $G/G/1$ queue, W_n has mean $\mu_n = n\,\mathsf{E}[S]$ and $\sigma_n^2 = n\,\mathsf{V}[S]$.

h.3.3.1. As a start, the function $\sin(t)$ does not have 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.3.3.3. Remember that $\{X_k\}$ and $\{S_k\}$ are sequences of i.i.d. random variables. What are the implications for the expectations?

h.3.3.4. What is the rate in, and what is the service capacity?

h.3.4.1. Consider a queueing system with constant service and constant inter-arrival times.

h.4.1.1. Think about the inter-departure time of a machine that produces items every 10 minutes. What inter-arrival times will a down-stream machine see?

h.4.1.4. What is λ? What is C_a^2 in the $G/G/1$ setting; what is it in the $M/G/1$ setting?

h.4.1.5. Why is $\lambda\beta$ the rate at which jobs are lost? What, then, is the rate at which jobs are accepted to the system? Observe that $\mu\,\mathsf{E}[L_s]$ is the rate at which jobs leave the system.

h.9. Let S be the processing (or service) time at the server, and S_i the service time of a type i job. Then,

$$S = \mathbb{1}_{T=1} S_1 + \mathbb{1}_{T=2} S_2. \tag{7.4.12}$$

h.4.2.2. An arbitrary job has to wait half the time it takes to form a batch.

h.4.2.3. Use (1.2.5) and (1.2.6).

h.4.3.1. Get the units right. Compute the load, and then the rest.

h.4.3.2. Realize that we now deal with setups and batch processing.

h.4.3.3. If there is no interruption, the service time is S_0. However, if there is an interruption, the service of a job is $R + S_0$.

h.4.3.5. First compute $\mathsf{E}\left[S^2\right]$. See [4.3.4].

h.4.4.1. Mind to work in a consistent set of units, e.g., hours. It is easy to make mistakes.

h.4.4.2. Observe that $m_f = 1/\lambda_f$ and $m_r = \mathsf{E}[R]$.

h.4.4.3. Is it relevant that $R_1,\ldots,R_n$ are mutually independent?

h.4.4.4. Use (1.2.4) and [4.4.3].

h.4.4.5. The joint distribution of $S_0 = s, N = k$ is $g(s)e^{-\lambda s}(\lambda s)^k/k!$. Use [2.2.8].

h.4.4.6. Realize that $\mathsf{E}[N] = \lambda_f \mathsf{E}[S_0]$. Then use [4.4.2].

h.4.4.9. Use (1.2.4) and [4.4.5].

h.4.4.10. Use Wald's equation, which we derived in [4.4.4].

h.4.4.14. Just realize that $\mathsf{E}[S] = \mathsf{E}[S_0]/A$, and use the above.

h.5.1.1. Observe that $Y(t) = \int_0^t \mathbb{1}_{L(s)>0}\,\mathrm{d}s$ is the total amount of time the server has been busy up to the time t. Then take $T_k = D_k$ as the epochs at which to inspect $Y(t)$, and finally use (3.3.4), i.e., rate stability.

h.5.1.2.

$$\frac{1}{t}\sum_{k=1}^{A(t)} S_k \geq \frac{1}{t}\int_0^t \mathbb{1}_{L(s)>0}\,\mathrm{d}s \geq \frac{1}{t}\sum_{k=1}^{D(t)} S_k. \tag{7.4.13}$$

h.5.3.3. Use that $\lambda \geq \delta$ always holds. Thus, when $\lambda \neq \delta$, it must be that $\lambda > \delta$. What are the consequences of this inequality; how does the queue length behave as a function of time?

h.5.3.4. Check that the conditions of the renewal reward theorem are satisfied in the above proof of (5.3.1b). Then define

$$Y(t) := A(n,t) = \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=n} \tag{7.4.14}$$

$$X_k := Y(A_k) - Y(A_{k-1}) = A(n, A_k) - A(n, A_{k-1}) = \mathbb{1}_{L(A_k-)=n}. \tag{7.4.15}$$

h.5.4.5. Substitute the definition of $L(s)$ in the LHS, then reverse the integral and summation.

h.6.1.1. First show that $M_L(s) = (1-\rho)\sum_n e^{sn}\rho^n$, then use (1.2.3d). Similarly, $\mathsf{P}\{L \geq n\} = \sum_{k\geq n} p(k)$.

h.6.1.5. Fill in $c = 1$. Realize that this is a check on the formulas.

h.6.1.6. Realize that the $M/M/c/c$ queue is similar to the $M/M/c$ queue. However, there cannot be more than c jobs in the system.

h.6.1.7. Use that for any x, $x^n/n! \to 0$ as $n \to \infty$.

h.6.1.10. 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 $n \to \infty$?

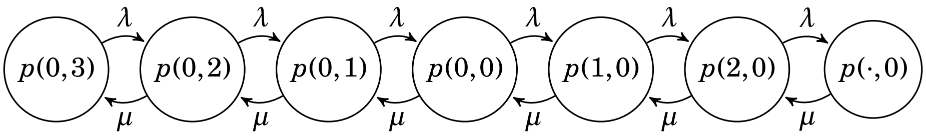
h.6.2.1. $\mathsf{E}[Q]$ follows right away from an application of Little's law. For the other quantities we need to find $\mathsf{E}[S]$. Use the expression for $\mathsf{E}[W(M/M/1)]$ to solve for ρ. Then, since λ is known, $\mathsf{E}[S]$ follows.

h.6.2.3. This is an $M/M/1/3$ queue; there is room for 1 customer in service and two in queue.

h.6.2.4. This is a queue with balking.

h.6.2.5. We are dealing with an $M/M/c$ queue. And, what is the implication of the remark about the compensation rate?

h.6.2.6. Let p_{ij} be the fraction of time that the system contains i riders and j taxi cabs. When a group arrives, they take a taxi, so the number of taxis decreases by one. If there are no taxis, the group has to wait. When a new taxi arrives, the number of groups is reduced by one, and so on, until there are 3 taxis waiting and no groups of people. Thus group arrivals acts as job arrivals, and taxi arrivals as services. Here is an overview of the transitions where μ is the rate at which cabs arrive, and λ is the arrival rate of parties of riders.



Now map this system to an $M/M/1$ queue.

h.6.3.2. Use (6.3.3) and that $\mathsf{V}[B] \geq 0$.

h.6.3.3. $\mathsf{P}\{B = k\} = q^{k-1}p$ with $q = 1 - p$. Use generating functions to compute $\mathsf{E}[B]$ and $\mathsf{E}[B^2]$.

h.6.3.4. Suppose the size of the kth batch is $B_k = 2$. Writing S_1 for the service time of the first item of this batch and S_2 for the service of the second item, $X_k = 2S_1 + 1S_2$. Now consider general batch sizes B, but realize that B is a random number, hence, the sun is over a random number of random variables.

h.6.3.7. Use [1.2.6] and [1.2.7].

h.6.4.2. Use that $\mathsf{V}[S] = 0$ for the $M/D/1$ queue.

h.6.4.4. The rate of accepted jobs is $\lambda\pi(0)$, hence $\rho = \lambda\pi(0)\mathsf{E}[S]$. But also $\rho = 1 - \pi(0)$. Now solve for $\pi(0)$.

h.6.4.5. Inter-arrivals times are memoryless for the $M/G/1$; for $\mathsf{E}[U]$ use the renewal-reward theorem to see that $\rho = \mathsf{E}[U]/(\mathsf{E}[I] + \mathsf{E}[U])$ for the $G/G/1$.

h.6.4.7. Use [6.3.6].

h.6.4.8. Use the PASTA property.

h.6.4.9. Realize that when estimating $\mathsf{E}[S_r]$ along a sample path, $S_r = 0$ for jobs that arrive at an empty system.

h.6.4.10. Apply PASTA and (5.3.9).

h.6.5.2. Show first that

$$\mu \mathsf{E}[L] = \mu \sum_{n=0}^{\infty} n\pi(n) = \lambda \frac{\mathsf{E}[B^2]}{2} + \lambda \mathsf{E}[B]\mathsf{E}[L] + \lambda \frac{\mathsf{E}[B]}{2}. \tag{7.4.16}$$

h.6.6.2. Do [6.6.1] first.

h.6.6.3. Use [2.4.1] to simplify the integral, or use [2.2.13] and [2.4.14].

h.6.6.4. Use [6.6.3].

h.6.6.5. Solve [6.6.3] and [6.6.4] first. Use shorthands: $\alpha = \lambda/(\lambda+\mu) \implies \mu/(\lambda+\mu) = 1-\alpha \implies \alpha/(1-\alpha) = \lambda/\mu = \rho$.

h.7.1.1. Fill $V(q) = aq^2 + bq$ into (7.1.3). Match the coefficients of q^2, q.

h.7.1.2. What is the cost of one cycle? What is the duration of one cycle?

h.7.1.4. Use (1.2.3d).

h.7.2.2. Use the argumentation that leads to (5.1.3).

h.7.2.3.

$$aq^2 = aq^2, \tag{7.4.17}$$
$$bq = 2aq\,\mathsf{E}[Y] - 2aq + bq + hq\,\mathsf{E}[S], \tag{7.4.18}$$
$$0 = a\,\mathsf{E}\left[Y^2\right] - 2a\,\mathsf{E}[Y] + a + b\,\mathsf{E}[Y] - b + \frac{1}{2}\lambda h\,\mathsf{E}\left[S^2\right]. \tag{7.4.19}$$

h.7.2.4. Use [4.4.5] to see that $\mathsf{E}\left[Y^2\right] = \lambda^2 \mathsf{E}\left[S^2\right] + \lambda \mathsf{E}[S]$.

h.7.2.6. Take $K = 0$ and $N = 1$, and realize that the LHS is $h\,\mathsf{E}[L]$.

h.7.3.1. Conditioning on the server being idle or busy at a departure leads to $f_D(t) = f_{X+S}(t)\mathsf{P}\{\text{server is idle}\} + f_S(t)\mathsf{P}\{\text{ server is busy }\}$. Next, use [6.6.2].

h.7.3.2. What does it mean when $P_{i0} > 0$?

h.7.3.4. Realize that an arrival is required to leave state $(0,0)$, and a departure at the second queue is necessary to enter state $(0,0)$.

h.7.4.1. After visiting M stations, any job must have left the tandem network.

h.7.4.2. Define $Q = P^M$. As P is transient, $Q_{ij} < 1$ for all i,j. Then show that $Q^n \to 0$.

s.1.2.1. Take $f(x) = h^\alpha$. Then,

$$\lim_{h\to 0}\frac{f(h)}{h} = \lim_{h\to 0}\frac{h^\alpha}{h} = \lim_{h\to 0} h^{\alpha-1}. \tag{7.4.20}$$

s.1.2.2. In fact (1) is trivial: $|f(h)| \leq |f(h)/h|$ when $|h| < 1$. But it is given that the RHS goes to zero. For (2) and (3):

$$\lim_{h\to 0}\frac{cf(h)}{h} = c\lim_{h\to 0}\frac{f(h)}{h} = 0, \text{ as } f = o(h), \tag{7.4.21}$$

$$\lim_{h\to 0}\frac{f(h)+g(h)}{h} = \lim_{h\to 0}\frac{f(h)}{h} + \lim_{h\to 0}\frac{g(h)}{h} = 0. \tag{7.4.22}$$

For (4), use the Algebraic Limit Theorem and multiply with h/h,

$$\lim_{h\to 0}\frac{f(h)g(h)}{h} = \lim_{h\to 0} h\frac{f(h)}{h}\frac{g(h)}{h} = \lim_{h\to 0} h \lim_{h\to 0}\frac{f(h)}{h}\lim_{h\to 0}\frac{g(h)}{h} = 0 \tag{7.4.23}$$

s.1.2.3. When $|x| \ll 1$, the terms with $n \geq 2$ in (1.2.3c) are $x^n = o(x)$. Then applying $x^n + x^m = o(x)$ to the Taylor series gives the result.

s.1.2.4. To see (1.2.4a), note first that $X\,\mathbb{1}_{X=n} = n\,\mathbb{1}_{X=n}$ because $X = n$ when $\mathbb{1}_{X=n} = 1$, and second that $\sum_{n=0}^{\infty}\mathbb{1}_{X=n} = 1$, since X takes one of the values in $\mathbb{N}$, and events $\{X = n\}$ and $\{X = m\}$ are non-overlapping when $n \neq m$.

s.1.2.5. This is just rewriting the definition:

$$G(k) = \mathsf{P}\{X > k\} = \sum_{m=k+1}^{\infty}\mathsf{P}\{X = m\} = \sum_{m=k+1}^{\infty} f(m) = \sum_{m=0}^{\infty}\mathbb{1}_{m>k} f(m). \tag{7.4.24}$$

s.1.2.6. Observe first that $\sum_{k=0}^{\infty}\mathbb{1}_{m>k} = m$, since $\mathbb{1}_{m>k} = 1$ if $k < m$ and $\mathbb{1}_{m>k} = 0$ if $k \geq m$. With this,

$$\sum_{k=0}^{\infty} G(k) = \sum_{k=0}^{\infty}\mathsf{P}\{X > k\} = \sum_{k=0}^{\infty}\sum_{m=k+1}^{\infty}\mathsf{P}\{X = m\} \tag{7.4.25}$$

$$= \sum_{k=0}^{\infty}\sum_{m=0}^{\infty}\mathbb{1}_{m>k}\,\mathsf{P}\{X = m\} = \sum_{m=0}^{\infty}\sum_{k=0}^{\infty}\mathbb{1}_{m>k}\,\mathsf{P}\{X = m\} \tag{7.4.26}$$

$$= \sum_{m=0}^{\infty} m\,\mathsf{P}\{X = m\} = \mathsf{E}[X]. \tag{7.4.27}$$

There are some technical details with respect to the interchange of the summations. Since the summands are positive, this is allowed. For further detail, see Capiński and Zastawniak [2003].

s.1.2.7.

$$\sum_{i=0}^{\infty} iG(i) = \sum_{i=0}^{\infty} i \sum_{n=i+1}^{\infty} \mathsf{P}\{X=n\} = \sum_{n=0}^{\infty} \mathsf{P}\{X=n\} \sum_{i=0}^{\infty} i\, \mathbb{1}_{n\geq i+1} \tag{7.4.28}$$

$$= \sum_{n=0}^{\infty} \mathsf{P}\{X=n\} \sum_{i=0}^{n-1} i = \sum_{n=0}^{\infty} \mathsf{P}\{X=n\} \frac{(n-1)n}{2} \tag{7.4.29}$$

$$= \sum_{n=0}^{\infty} \frac{n^2}{2} \mathsf{P}\{X=n\} - \frac{\mathsf{E}[X]}{2} = \frac{\mathsf{E}[X^2]}{2} - \frac{\mathsf{E}[X]}{2}. \tag{7.4.30}$$

s.1.2.8.

$$\mathsf{E}[X] = \int_0^\infty x F(\mathrm{d}x) = \int_0^\infty \int_0^x \mathrm{d}y F(\mathrm{d}x) \tag{7.4.31}$$

$$= \int_0^\infty \int_0^\infty \mathbb{1}_{y\leq x}\, \mathrm{d}y F(\mathrm{d}x) = \int_0^\infty \int_0^\infty \mathbb{1}_{y\leq x} F(\mathrm{d}x)\,\mathrm{d}y \tag{7.4.32}$$

$$= \int_0^\infty \int_y^\infty F(\mathrm{d}x)\,\mathrm{d}y = \int_0^\infty G(y)\,\mathrm{d}y. \tag{7.4.33}$$

s.1.2.9.

$$\int_0^\infty yG(y)\,\mathrm{d}y = \int_0^\infty y \int_y^\infty f(x)\,\mathrm{d}x\,\mathrm{d}y = \int_0^\infty y \int_0^\infty \mathbb{1}_{y\leq x} f(x)\,\mathrm{d}x\,\mathrm{d}y \tag{7.4.34}$$

$$= \int_0^\infty f(x) \int_0^\infty y\, \mathbb{1}_{y\leq x}\,\mathrm{d}y\,\mathrm{d}x = \int_0^\infty f(x) \int_0^x y\,\mathrm{d}y\,\mathrm{d}x \tag{7.4.35}$$

$$= \int_0^\infty f(x) \frac{x^2}{2}\,\mathrm{d}x = \frac{\mathsf{E}[X^2]}{2}. \tag{7.4.36}$$

s.1.2.10. Use integration by parts.

$$\int_0^\infty yG(y)\,\mathrm{d}y = \frac{y^2}{2}G(y)\Big|_0^\infty - \int_0^\infty \frac{y^2}{2} g(y)\,\mathrm{d}y = \int_0^\infty \frac{y^2}{2} f(y)\,\mathrm{d}y = \frac{\mathsf{E}[X^2]}{2}, \tag{7.4.37}$$

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

s.1.2.11. $M_X(0) = \mathsf{E}[e^{0X}] = \mathsf{E}[e^0] = \mathsf{E}[1] = 1$.

s.1.2.12. The probability that the first child to guess 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

$$\begin{aligned}
&\mathsf{P}\{\text{Child 2 wins}\}\\
&= \mathsf{P}\{\text{Child 2 guesses the right number and child 1 does not win}\}\\
&= \mathsf{P}\{\text{Child 2 guesses the right number} \mid \text{child 1 does not win}\} \cdot \mathsf{P}\{\text{Child 1 does not win}\}\\
&= \mathsf{P}\{\text{Child 2 makes the right guess in the set } \{1,2\}\} \cdot \frac{2}{3}\\
&= \frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}.
\end{aligned} \tag{7.4.38}$$

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

s.2.1.1. $d_1 = 7$, $L_1 = 8-7+5 = 6$, $d_2 = 6$, $L_2 = 6-6+4 = 4$, $d_3 = 4$, $L_3 = 4-4+9 = 9$.

s.2.1.2.

$$d_k = \min\{L_{k-1} + a_k, c_k\}, \tag{7.4.39}$$

$$L_k = L_{k-1} + a_k - d_k = L_{k-1} + a_k - \min\{L_{k-1} + a_k, c_k\} \tag{7.4.40}$$

$$= \max\{L_{k-1} + a_k - c_k, 0\}. \tag{7.4.41}$$

s.2.1.3. Here is the python code.

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

>>> K = 8

>>> for k in range(1, len(a)):
...     d[k] = min(L[k - 1], c[k])
...     Lp = L[k - 1] + a[k] - d[k]  #  without loss
...     L[k] = min(Lp, K)  #  chop off at K
...     l[k] = Lp - L[k]  #  lost
...

>>> print(L)
[0, 8, 6, 7]
>>> print(l)
[0, 2, 0, 0]
>>> print(sum(l) / sum(a))  # fraction lost.
0.10526315789473684
```

s.2.1.4. Since a fraction p is faulty, a fraction $1-p$ can depart.

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

>>> L[0] = 2

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

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 $\lambda(1+p)$, where $\lambda = \lim_{n\to\infty} n^{-1}\sum_{k=1}^n a_k$?

s.2.1.5. In code:

```
>>> a = [0, 4, 8, 2, 1]
>>> c = [3] * len(a)
>>> L_R = [0] * len(a) # repair jobs
>>> d_R = [0] * len(a) # departing repair jobs
>>> L_N = [0] * len(a) # new jobs
>>> d_N = [0] * len(a) # departing new jobs
>>> p = 0.2

>>> L_R[0] = 2
>>> L_N[0] = 8

>>> for k in range(1, len(a)):
...     d_R[k] = min(L_R[k - 1], 2 * c[k])
...     c_N = c[k] - d_R[k] / 2 # capacity left for new jobs
...     d_N[k] = min(L_N[k - 1], c_N)
...     a_R = int(p * d_N[k - 1] + 0.5) # rounding
...     a_N = a[k]
...     L_R[k] = L_R[k - 1] + a_R - d_R[k]
...     L_N[k] = L_N[k - 1] + a_N - d_N[k]
...
>>> print(L_R)
[2, 0, 0, 1, 1]
>>> print(L_N)
[8, 10.0, 15.0, 14.0, 12.5]
```

s.2.1.6. $\sum_{k=1}^T (\beta c_k + h L_k)$.

s.2.1.7. Take $c_k = c\,\mathbb{1}_{L_{k-1}>=N}$.

s.2.1.8. $I_0 = 0$, $d_k = \min\{L_{k-1}, I_k c_k\}$, from which L_k and I_{k+1} follow, and so on. Next, observing that the machine switches on in period k iff $I_{k-1} = 0$ and $I_k = 1$, the cost is $\sum_{k=1}^T (\beta I_k + h L_k + K[I_k - I_{k-1}]^+)$.

s.2.1.9.

$$J_{-,k} = \min\left\{m : \sum_{i=k}^{k+m} c_i > L_{k-1}\right\}, \qquad J_{+,k} = \min\left\{m : \sum_{i=k}^{k+m} c_i \geq L_{k-1} + a_k\right\}. \tag{7.4.42}$$

With a while loop serves well to compute these bounds Supposing that we already computed $\{L_k\}$, then for given k,

```
J_min, tot_service = 0, 0
while tot_service <= L[k - 1]:
    tot_service += c[k + J_min]
```

```
    J_min += 1

J_min -= 1 #  We ran one too far
```

s.2.1.10.

$$d_k^1 = \min\{L_{k-1}^1, c_k\}, \quad c_k^2 = c_k - d_k^1, \quad d_k^2 = \min\{L_{k-1}^2, c_k^2\}, \quad L_k^i = L_{k-1}^i + a_k^i - d_k^i. \tag{7.4.43}$$

s.2.1.11. Let c_k^i be the capacity allocated to queue i in period k. The fair rule gives that

$$c_k^1 = \operatorname{round}\frac{L_{k-1}^1}{L_{k-1}^1 + L_{k-1}^2}, \qquad c_k^2 = c_k - c_k^1, \tag{7.4.44}$$

$$d_k^i = \min\{L_{k-1}^i, c_k^i\}, \qquad L_k^i = L_{k-1}^i + a_k^i - d_k^i. \tag{7.4.45}$$

In general, rules to distribute capacity c_k over the queues can be based on game-theoretic ideas, such as the principle of *equal division of the contested sum*, see the work of Aumann and Maschler.

s.2.1.12. Queue 2 minimally needs $c_k^2 = \min\{L_{k-1}^2, r^2\}$, and with this,

$$d_k^1 = \min\{L_{k-1}^1, c_k - c_k^2\}, \qquad d_k^2 = \min\{L_{k-1}^2, c_k - d_k^1\}. \tag{7.4.46}$$

s.2.1.13. Queue 1 uses all capacity except r^2, queue 2 gets the left-over:

$$d_k^1 = \min\{L_{k-1}^1, c_k - r^2\}, \tag{7.4.47}$$

$$c_k^2 = \min\{c_k - r^1, c_k - d^1 k\} = c_k - \max\{r^1, d_k^1\}, \tag{7.4.48}$$

$$d_k^2 = \min\{L_{k-1}^2, c_k^2\}. \tag{7.4.49}$$

s.2.1.14. Let a_k^1 be the external arrivals at station A, and the departures of station 1 are the arrivals at station 2: $a_k^2 = d_k^1$. Thus,

$$d_k^i = \min\{L_{k-1}^i, c_k^i\}, \qquad L_k^i = L_{k-1}^i - d_k^i + a_k^i. \tag{7.4.50}$$

```python
a1 = [0, 2, 3, 8, 0, 9]
c1 = [3] * len(a1)
c2 = [2] * len(a1)
L1 = [0] * len(a1)
L2 = [0] * len(a1)

for k in range(1, len(a1)):
    d1 = min(L1[k], c1[k])
    L1[k] = L1[k - 1] + a1[k] - d1
    d2 = min(L2[k], c2[k])
    L2[k] = L2[k - 1] + d1 - d2
```

s.2.1.15. Using [2.1.13] with $d_k^1 = \min\{L_{k-1}^1, c_k^1, M - L_{k-1}^2\}$ is nearly correct. But, what if $L_{k-1}^2 > M$ for the first few periods? Therefore, take instead $d_k^1 = \min\{L_{k-1}^1, c_k^1, [M - L_{k-1}^2]^+\}$.

```python
a1 = [0, 2, 3, 8, 0, 9]
c1 = [3] * len(a1)
c2 = [2] * len(a1)
L1 = [0] * len(a1)
L2 = [0] * len(a1)

M2 = 10

if L2[0] > M2:
    print("The starting value of L2 is too large")
    exit(0)

for k in range(1, len(a1)):
    d1 = min(L1[k], c1[k], M2 - L2[k - 1])
    L1[k] = L1[k - 1] + a1[k] - d1
    d2 = min(L2[k], c2[k])
    L2[k] = L2[k - 1] + d1 - d2
```

s.2.1.16. Take $a_k^C = d_k^A + d_k^B$ and use (2.1.1) for each station.

s.2.1.17. At the mixing machine $d_k = \min\{L_k, c_k\}$. Therefore, in this very simple model, $a_k^A = d_k \lambda^A/(\lambda^A + \lambda^B)$. Now use [2.1.14].

s.2.1.18. With $d_k^i = \min\{L_{k-1}^i, c_k^i \mathbb{1}_{p_k=i}\}$ we can specify the evolution of queue i. So it remains to deal with p_{k+1}. For this, we use the following 'truth table'. This can be implemented in code to specify what p_{k+1} has to become.

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

Here is an important warning: in such tables it is very, very easy to miss one (or more) cases. Here, each of the indicators has 2 possible values, and p_k has 3, so there are $2 \times 2 \times 3 = 12$ cases, all of which are covered in the table.

s.2.2.1.

$$\mathsf{E}[N_n(t)] = \mathsf{E}\left[\sum_{i=1}^n B_i\right] = \sum_{i=1}^n \mathsf{E}[B_i] = n\,\mathsf{E}[B_i] = np. \tag{7.4.51}$$

s.2.2.2.

$$\binom{n}{k}\left(\frac{\lambda t}{n}\right)^k\left(1-\frac{\lambda t}{n}\right)^{n-k} = \frac{n!}{k!(n-k)!}\left(\frac{\lambda t}{n}\frac{n}{n-\lambda t}\right)^k\left(1-\frac{\lambda t}{n}\right)^n \tag{7.4.52}$$

$$= \frac{(\lambda t)^k}{k!}\left(\frac{n}{n-\lambda t}\right)^k\frac{n!}{n^k(n-k)!}\left(1-\frac{\lambda t}{n}\right)^n \tag{7.4.53}$$

$$= \frac{(\lambda t)^k}{k!}\left(\frac{n}{n-\lambda t}\right)^k\frac{n}{n}\frac{n-1}{n}\cdots\frac{n-k+1}{n}\left(1-\frac{\lambda t}{n}\right)^n. \tag{7.4.54}$$

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, use (1.2.3b) to see that $\left(1-\frac{\lambda t}{n}\right)^n\to e^{-\lambda t}$.

s.2.2.3. $N_n(t)$ is a binomially distributed random variable with parameters n and p. The maximum value of $N_n(t)$ is n. The random variable $N(t)$ models the number of arrivals that can occur during $[0,t]$. As such it is not necessarily bounded by n. Thus, $N_n(t)$ and $N(t)$ cannot represent the same random variable.

s.2.2.4. Write $N(s,t]$ for the number of arrivals in the interval $(s,t]$. First we make a few simple observations: $N(t+\Delta t] = N(t)+N(t,t+\Delta t]$, hence

$$\mathbb{1}_{N(t+\Delta t)=n,N(t)=n} = \mathbb{1}_{N(t)+N(t,t+\Delta t]=n,N(t)=n} = \mathbb{1}_{N(t,t+\Delta t]=0,N(t)=n}. \tag{7.4.55}$$

Thus,

$$\mathsf{P}\{N(t+\Delta t)=n\,|\,N(t)=n\} = \frac{\mathsf{P}\{N(t+\Delta t)=n,N(t)=n\}}{\mathsf{P}\{N(t)=n\}} \tag{7.4.56}$$

$$= \frac{\mathsf{P}\{N(t,t+\Delta t]=0,N(t)=n\}}{\mathsf{P}\{N(t)=n\}} \tag{7.4.57}$$

$$= \frac{\mathsf{P}\{N(t,t+\Delta t]=0\}\,\mathsf{P}\{N(t)=n\}}{\mathsf{P}\{N(t)=n\}} \quad \text{(independence)} \tag{7.4.58}$$

$$= \mathsf{P}\{N(t,t+\Delta t]=0\} = \mathsf{P}\{N(0,\Delta t]=0\} \quad \text{(stationarity)} \tag{7.4.59}$$

$$= e^{-\lambda\Delta t}(\lambda\Delta t)^0/0! = e^{-\lambda\Delta t} = 1-\lambda\Delta t+o(\Delta t). \tag{7.4.60}$$

s.2.2.5.

$$\mathsf{P}\{N(t+\Delta t)=n+1\,|\,N(t)=n\} = \frac{\mathsf{P}\{N(t+\Delta t)=n+1,N(t)=n\}}{PN(t)=n} \tag{7.4.61}$$

$$= \mathsf{P}\{N(t,t+\Delta t]=1\} = e^{-\lambda\Delta t}\frac{(\lambda\Delta t)^1}{1!} \tag{7.4.62}$$

$$= (1-\lambda\Delta t+o(\Delta t))\lambda\Delta t = \lambda\Delta t-\lambda^2\Delta t^2+o(\Delta t) \tag{7.4.63}$$

$$= \lambda\Delta t+o(\Delta t). \tag{7.4.64}$$

s.2.2.6.

$$\mathsf{P}\{N(t+\Delta t)\geq n+2\,|\,N(t)=n\} = \mathsf{P}\{N(t,t+\Delta t]\geq 2\} \tag{7.4.65}$$

$$= e^{-\lambda\Delta t}\sum_{i=2}^{\infty}\frac{(\lambda\Delta t)^i}{i!} = e^{-\lambda\Delta t}\left(\sum_{i=0}^{\infty}\frac{(\lambda\Delta t)^i}{i!}-\lambda\Delta t-1\right) \tag{7.4.66}$$

$$= e^{-\lambda\Delta t}(e^{\lambda\Delta t}-1-\lambda\Delta t) = 1-e^{-\lambda\Delta t}(1+\lambda\Delta t) \tag{7.4.67}$$

$$= 1-(1-\lambda\Delta t+o(\Delta t))(1+\lambda\Delta t) = 1-(1-\lambda^2\Delta t^2+o(\Delta t)) \tag{7.4.68}$$

$$= \lambda^2\Delta t^2+o(\Delta t) = o(\Delta t), \tag{7.4.69}$$

where we expand

$$(1-\lambda\Delta t+o(\Delta t))(1+\lambda\Delta t)=1+\lambda\Delta t-\lambda\Delta t-\lambda^2\Delta^2t^2+o(\Delta t). \tag{7.4.70}$$

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

$$\begin{aligned}
\mathsf{P}\{N(t+\Delta t)\geq n+2\,|\,N(t)=n\} &= \mathsf{P}\{N(t,t+\Delta t]\geq 2\}=1-\mathsf{P}\{N(t,t+\Delta t]<2\} && (7.4.71)\\
&= 1-\mathsf{P}\{N(t,t+\Delta t]=0\}-\mathsf{P}\{N(t,t+\Delta t]=1\} && (7.4.72)\\
&= 1-(1-\lambda\Delta t+o(\Delta t))-(\lambda\Delta t+o(\Delta t)) && (7.4.73)\\
&= o(\Delta t). && (7.4.74)
\end{aligned}$$

s.2.2.7. When a random variable N is Poisson distributed with parameter λt,

$$\begin{aligned}
\mathsf{E}[N] &= \sum_{n=0}^{\infty} ne^{-\lambda t}\frac{(\lambda t)^n}{n!}=\sum_{n=1}^{\infty} ne^{-\lambda t}\frac{(\lambda t)^n}{n!}=e^{-\lambda t}\lambda t\sum_{n=1}^{\infty}\frac{(\lambda t)^{n-1}}{(n-1)!} && (7.4.75)\\
&= e^{-\lambda t}\lambda t\sum_{n=0}^{\infty}\frac{(\lambda t)^n}{n!}=e^{-\lambda t}\lambda te^{\lambda t}=\lambda t. && (7.4.76)
\end{aligned}$$

s.2.2.8.

$$\begin{aligned}
\mathsf{E}\left[N^2\right] &= \sum_{n=0}^{\infty} n^2e^{-\lambda t}\frac{(\lambda t)^n}{n!}=e^{-\lambda t}\sum_{n=1}^{\infty} n\frac{(\lambda t)^n}{(n-1)!}=e^{-\lambda t}\sum_{n=0}^{\infty}(n+1)\frac{(\lambda t)^{n+1}}{n!} && (7.4.77)\\
&= e^{-\lambda t}\lambda t\sum_{n=0}^{\infty} n\frac{(\lambda t)^n}{n!}+e^{-\lambda t}\lambda t\sum_{n=0}^{\infty}\frac{(\lambda t)^n}{n!}=(\lambda t)^2+\lambda t. && (7.4.78)
\end{aligned}$$

s.2.2.9. $\mathsf{V}[N]=\mathsf{E}\left[N^2\right]-(\mathsf{E}[N])^2=(\lambda t)^2+\lambda t-(\lambda t)^2=\lambda t.$

s.2.2.10. From the hint,

$$\begin{aligned}
\mathsf{P}\{N(0,s]=1\,|\,N(0,t]=1\} &= \frac{\mathsf{P}\{N(0,s]=1,N(0,t]=1\}}{\mathsf{P}\{N(0,t]=1\}}=\frac{\mathsf{P}\{N(0,s]=1,N(s,t]=0\}}{\mathsf{P}\{N(0,t]=1\}} && (7.4.79)\\
&= \frac{\mathsf{P}\{N(0,s]=1\}\,\mathsf{P}\{N(s,t]=0\}}{\mathsf{P}\{N(0,t]=1\}}=\frac{\lambda se^{-\lambda s}e^{-\lambda(t-s)}}{\lambda te^{-\lambda t}}=\frac{s}{t}. && (7.4.80)
\end{aligned}$$

s.2.2.11.

$$SCV=\frac{\mathsf{V}[N(t)]}{(\mathsf{E}[N(t)])^2}=\frac{\lambda t}{(\lambda t)^2}=\frac{1}{\lambda t}. \tag{7.4.81}$$

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

s.2.2.12.

$$\begin{aligned}
\mathsf{P}\left\{N_\lambda(t)+N_\mu(t)=n\right\} &= \sum_{i=0}^{n}\mathsf{P}\left\{N_\mu(t)=n-i\right\}\mathsf{P}\{N_\lambda(t)=i\} && (7.4.82)\\
&= \sum_{i=0}^{n}\frac{(\mu t)^{n-i}}{(n-i)!}\frac{(\lambda t)^i}{i!}e^{-(\mu+\lambda)t}=e^{-(\mu+\lambda)t}\sum_{i=0}^{n}\frac{(\mu t)^{n-i}}{(n-i)!}\frac{(\lambda t)^i}{i!} && (7.4.83)\\
&= e^{-(\mu+\lambda)t}\frac{1}{n!}\sum_{i=0}^{n}\binom{n}{i}(\mu t)^{n-i}(\lambda t)^i \quad \text{(binomial formula)} && (7.4.84)\\
&= \frac{((\mu+\lambda)t)^n}{n!}e^{-(\mu+\lambda)t}. && (7.4.85)
\end{aligned}$$

s.2.2.13. With the above:

$$\mathsf{P}\{N_\lambda(t)=1 \,|\, N_\lambda(t)+N_\mu(t)=1\} = \frac{\mathsf{P}\{N_\lambda(t)=1, N_\lambda(t)+N_\mu(t)=1\}}{\mathsf{P}\{N_\lambda(t)+N_\mu(t)=1\}} \tag{7.4.86}$$

$$= \frac{\mathsf{P}\{N_\lambda(t)=1, N_\mu(t)=0\}}{\mathsf{P}\{N_{\lambda+\mu}(t)=1\}} = \frac{\mathsf{P}\{N_\lambda(t)=1\}\,\mathsf{P}\{N_\mu(t)=0\}}{\mathsf{P}\{N_{\lambda+\mu}(t)=1\}} \tag{7.4.87}$$

$$= \frac{\lambda t \exp(-\lambda t)\exp(-\mu t)}{((\lambda+\mu)t)\exp(-(\lambda+\mu)t)} = \frac{\lambda t \exp(-(\lambda+\mu)t)}{((\lambda+\mu)t)\exp(-(\lambda+\mu)t)} = \frac{\lambda}{\lambda+\mu}. \tag{7.4.88}$$

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

s.2.2.14.

$$\mathsf{P}\{N_1(t)=k\} = \sum_{n=k}^\infty \mathsf{P}\{N_1(t)=k, N(t)=n\} = \sum_{n=k}^\infty \mathsf{P}\{N_1(t)=k \,|\, N(t)=n\}\,\mathsf{P}\{N(t)=n\} \tag{7.4.89}$$

$$= \sum_{n=k}^\infty \mathsf{P}\{N_1(t)=k \,|\, N(t)=n\} e^{-\lambda t}\frac{(\lambda t)^n}{n!} \tag{7.4.90}$$

$$= \sum_{n=k}^\infty \binom{n}{k} p^k (1-p)^{n-k} e^{-\lambda t}\frac{(\lambda t)^n}{n!}, \quad \text{by the hint} \tag{7.4.91}$$

$$= e^{-\lambda t}\sum_{n=k}^\infty \frac{p^k(1-p)^{n-k}}{k!(n-k)!}(\lambda t)^n = e^{-\lambda t}\frac{(\lambda t p)^k}{k!}\sum_{n=k}^\infty \frac{(\lambda t(1-p))^{n-k}}{(n-k)!} \tag{7.4.92}$$

$$= e^{-\lambda t}\frac{(\lambda t p)^k}{k!}\sum_{n=0}^\infty \frac{(\lambda t(1-p))^n}{n!} = e^{-\lambda t}\frac{(\lambda t p)^k}{k!} e^{\lambda t(1-p)} = e^{-\lambda t p}\frac{(\lambda t p)^k}{k!}. \tag{7.4.93}$$

s.2.2.15. Since $N(t)$ is Poisson distributed with parameter λt,

$$M_{N(t)}(s) = \mathsf{E}\left[e^{sN(t)}\right] = \sum_{k=0}^\infty e^{sk}\,\mathsf{P}\{N(t)=k\} = \sum_{k=0}^\infty e^{sk}\frac{(\lambda t)^k}{k!}e^{-\lambda t} \tag{7.4.94}$$

$$= e^{-\lambda t}\sum_{k=0}^\infty \frac{(e^s\lambda t)^k}{k!} = \exp(-\lambda t + e^s\lambda t) = \exp(\lambda t(e^s-1)). \tag{7.4.95}$$

s.2.2.16. Using the expression for the moment-generating function of [2.2.15],

$$M'_{N(t)}(s) = \lambda t e^s \exp(\lambda t(e^s-1)). \tag{7.4.96}$$

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

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

$$\mathsf{E}\left[e^{sZ}\right] = e^0\,\mathsf{P}\{Z=0\} + e^s\,\mathsf{P}\{Z=1\} = (1-p) + e^s p. \tag{7.4.97}$$

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

$$\mathsf{E}\left[e^{s\sum_{i=1}^n Z_i}\right] = \left(\mathsf{E}\left[e^{sZ}\right]\right)^n = \left(1+p(e^s-1)\right)^n, \tag{7.4.98}$$

where we use (1.2.7d).

With (1.2.4a),

$$\mathsf{E}\left[e^{sY}\right] = \mathsf{E}\left[\sum_{n=0}^{\infty} e^{s\sum_{i=1}^{N} Z_i}\,\mathbb{1}_{N=n}\right] = \mathsf{E}\left[\sum_{n=0}^{\infty} e^{s\sum_{i=1}^{n} Z_i}\,\mathbb{1}_{N=n}\right] = \sum_{n=0}^{\infty} \mathsf{E}\left[e^{s\sum_{i=1}^{n} Z_i}\,\mathbb{1}_{N=n}\right] \tag{7.4.99}$$

$$= \sum_{n=0}^{\infty} \mathsf{E}\left[e^{s\sum_{i=1}^{n} Z_i}\right] \mathsf{E}\left[\mathbb{1}_{N=n}\right], \quad \text{by independence of } Z_i \text{ and } N, \tag{7.4.100}$$

$$= \sum_{n=0}^{\infty} \left(1+p(e^s-1)\right)^n \mathsf{P}\{N=n\} \tag{7.4.101}$$

$$= \sum_{n=0}^{\infty} \left(1+p(e^s-1)\right)^n e^{-\lambda}\frac{\lambda^n}{n!} = e^{-\lambda}\sum_{n=0}^{\infty}\frac{(1+p(e^s-1))^n\,\lambda^n}{n!} \tag{7.4.102}$$

$$= e^{-\lambda}\exp(\lambda(1+p(e^s-1))) = \exp(\lambda p(e^s-1)). \tag{7.4.103}$$

s.2.2.18.

$$M_{N_\lambda(t)+N_\mu(t)}(s) = M_{N_\lambda(t)}(s)\cdot M_{N_\mu(t)}(s) = \exp(\lambda t(e^s-1))\cdot\exp(\mu t(e^s-1)) \tag{7.4.104}$$

$$= \exp((\lambda+\mu)t(e^s-1)). \tag{7.4.105}$$

s.2.2.19. Take $Y = \sum_{i=1}^{n} Z_i$ with $Z_i \sim B(p)$. Then,

$$M_Y(s) = \mathsf{E}\left[e^{s\sum_{i=1}^{n} Z_i}\right] = \left(\mathsf{E}\left[e^{sZ}\right]\right)^n = \left(1+p(e^s-1)\right)^n. \tag{7.4.106}$$

Recall that $p = \lambda t/n$. Then, with (1.2.3b),

$$\lim_{n\to\infty}\left(1+\frac{\lambda t}{n}(e^s-1)\right)^n = \exp(\lambda t(e^s-1)). \tag{7.4.107}$$

s.2.3.1. For every $A_k \leq t$, we have that $\mathbb{1}_{A_k\leq t} = 1$, and else the indicator is 0. Hence, in the summation we count the number of times $A_k \leq t$.

s.2.3.2. Yes, it is true.

s.2.3.3. $A(t)$ 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 A_n, i.e., $A(A_n)$, must be n.

s.2.3.4. Suppose $A_3 = 10$ and $A_4 = 20$. Take $t = 15$. Then $\min\{k : A_k \geq 15\} = 4$ since $A_3 < t = 15 < A_4$. On the other hand $\max\{k : A_k \leq t\} = 3$. And, indeed, at time $t = 15$, 3 jobs arrived, not 4. So defining $A(t)$ as $\min\{k : A_k \geq t\}$ is not OK. This example also shows that in general $A(t) \neq \min\{k : A_k > t\}$. So, neither definition is correct.

s.2.3.5. Let's feed it to the computer. Mind that in Python (just like in C, and so on), arrays start at index 0, not at index 1.

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

```
...     A[i] = A[i - 1] + X[i]
...
>>> A
[0, 10, 15, 21]

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

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

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

s.2.3.6. $A_0 = 0$, $A_1 = 10$, $A_2 = 20$, and so on. Hence, $A_k = 10k$. $W_0 = 0$, $W_1 = \max\{0+0-10, 0\} = 0$. $W_2 = \max\{0+11-10, 0\} = 1$. $W_3 = \max\{1+11-10, 0\} = 2$. Hence, $W_k = k-1$ for $k \geq 1$. Thus, $W_k = k-1+11 = k+10$ for $k \geq 1$, and $D_k = 10k + k + 10 = 11k + 10$. Note that W_k increases linearly as a function of k. All in all, $A(t) = \lfloor t/10 \rfloor$, and $D(t) = \lfloor (t-10)/11 \rfloor$.

s.2.3.7. First find the distribution of $Y_k := S_{k-1} - X_k$ so that we can write $W_k = [W_{k-1} + Y_k]^+$. Use independence of $\{S_k\}$ and $\{X_k\}$:

$$\mathsf{P}\{Y_k = -2\} = \mathsf{P}\{S_{k-1} - X_k = -2\} = \mathsf{P}\{S_{k-1} = 1, X_k = 3\} = \mathsf{P}\{S_{k-1} = 1\}\,\mathsf{P}\{X_k = 3\} = \frac{1}{4}. \tag{7.4.108}$$

Dropping the dependence on k for ease, we get

$$\mathsf{P}\{Y = -2\} = \mathsf{P}\{S - X = -2\} = \mathsf{P}\{S = 1, X = 3\} = \mathsf{P}\{S = 1\}\,\mathsf{P}\{X = 3\} = \frac{1}{4}, \tag{7.4.109}$$

$$\mathsf{P}\{Y = -1\} = \mathsf{P}\{S = 2\}\,\mathsf{P}\{X = 3\} = \frac{1}{4}, \tag{7.4.110}$$

$$\mathsf{P}\{Y = 0\} = \mathsf{P}\{S = 1\}\,\mathsf{P}\{X = 1\} = \frac{1}{4}, \tag{7.4.111}$$

$$\mathsf{P}\{Y = 1\} = \mathsf{P}\{S = 2\}\,\mathsf{P}\{X = 1\} = \frac{1}{4}. \tag{7.4.112}$$

With this

$$\mathsf{P}\{W_1 = 1\} = \mathsf{P}\{W_0 + Y = 1\} = \mathsf{P}\{3 + Y = 1\} = \mathsf{P}\{Y = -2\} = \frac{1}{4}, \tag{7.4.113}$$

$$\mathsf{P}\{W_1 = 2\} = \mathsf{P}\{3 + Y = 2\} = \mathsf{P}\{Y = -1\} = \frac{1}{4}, \tag{7.4.114}$$

$$\mathsf{P}\{W_1 = 3\} = \mathsf{P}\{3 + Y = 3\} = \mathsf{P}\{Y = 0\} = \frac{1}{4}, \tag{7.4.115}$$

$$\mathsf{P}\{W_1 = 4\} = \mathsf{P}\{3 + Y = 4\} = \mathsf{P}\{Y = 1\} = \frac{1}{4}. \tag{7.4.116}$$

$$\tag{7.4.117}$$

And, then

$$\begin{aligned}\mathsf{P}\{W_2 = 1\} = \mathsf{P}\{W_1 + Y = 1\} &= \sum_{i=1}^{4} \mathsf{P}\{W_1 + Y = 1 \,|\, W_1 = i\}\, \mathsf{P}\{W_1 = i\} \\ &= \sum_{i=1}^{4} \mathsf{P}\{i + Y = 1 \,|\, W_1 = i\} \frac{1}{4} = \sum_{i=1}^{4} \mathsf{P}\{Y = 1 - i \,|\, W_1 = i\} \frac{1}{4} \\ &= \frac{1}{4} \sum_{i=1}^{4} \mathsf{P}\{Y = 1 - i\} = \frac{1}{4}(\mathsf{P}\{Y = 0\} + \mathsf{P}\{Y = -1\} + \mathsf{P}\{Y = -2\}) = \frac{3}{16}.\end{aligned} \tag{7.4.118}$$

s.2.3.8. 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.2.3.9. 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,

$$V(t) = [V(A_{A(t)}) - (t - A_{A(t)})]^+ = [W_{A(t)} + (A_{A(t)} - t)]^+. \tag{7.4.119}$$

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

$$L(A_k-) = k - 1 - D(A_k-) = k - 1 - D(10k-) = k - 1 - \left\lfloor \frac{(10k-) - 10}{11} \right\rfloor. \tag{7.4.120}$$

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

s.2.3.11. We defined $\tilde{A}(t)$ as the amount of people up to time t that left the queue and moved to the server. It is then clear to see that the number of jobs in queue $Q(t)$ is equal to the number amount of jobs that have arrived, i.e., $A(t)$, minus the number of jobs that left the queue, i.e., $\tilde{A}(t)$. Using the same reasoning for $L_s(t)$, the second line also follows.

s.2.3.12. 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.2.3.13.

$$L(t) = A(t) - D(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t} - \sum_{k=1}^{\infty} \mathbb{1}_{D_k \le t} = \sum_{k=1}^{\infty} [\mathbb{1}_{A_k \le t} - \mathbb{1}_{D_k \le t}]. \tag{7.4.121}$$

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

$$L(t) = \sum_{k=1}^{\infty} [\mathbb{1}_{A_k \le t} - \mathbb{1}_{D_k \le t}] = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t < D_k}. \tag{7.4.122}$$

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

$$\begin{aligned} L(A_k) > 0 &\Leftrightarrow A(A_k) - D(A_k) > 0 \Leftrightarrow k - D(A_k) > 0 \Leftrightarrow k > D(A_k) \\ &\Leftrightarrow k - 1 \ge D(A_k) \Leftrightarrow D(D_{k-1}) \ge D(A_k) \Leftrightarrow D_{k-1} \ge A_k, \end{aligned} \tag{7.4.123}$$

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

s.2.3.15. 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,

$$L(A_k-) = L(A_{k-1}-) + 1 - \sum_{i=k-1-L(A_{k-1}-)}^{k-1} \mathbb{1}_{D_i < A_k}. \tag{7.4.124}$$

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

s.2.3.16. 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.

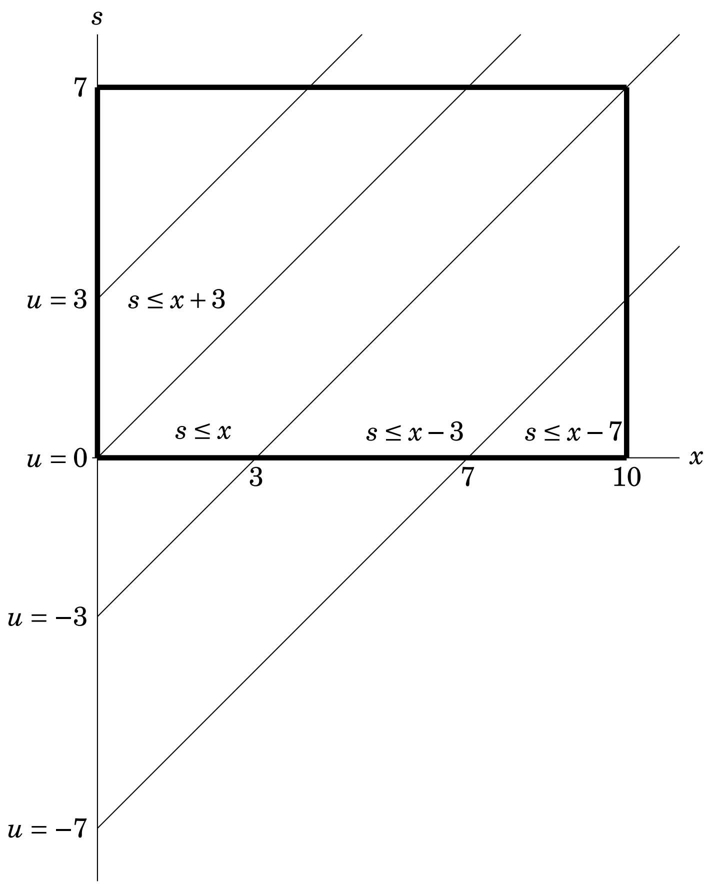
s.2.3.17. The joint density of S and X is given by

$$f_{XS}(x,s) = f_X(x)\cdot f_S(s) = \frac{1}{10}\mathbb{1}_{0\leq x\leq 10}\cdot\frac{1}{7}\mathbb{1}_{0\leq s\leq 7}, \tag{7.4.125}$$

since X and S are independent. Thus,

$$\begin{aligned} \mathsf{P}\{S-X\leq u\} = \mathsf{E}[\mathbb{1}_{S-X\leq u}] &= \frac{1}{70}\int_0^{10}\int_0^7 \mathbb{1}_{s-x\leq u}\,\mathrm{d}s\,\mathrm{d}x \\ &= \frac{1}{70}\int_0^{10}\int_0^7 \mathbb{1}_{s\leq x+u}\,\mathrm{d}s\,\mathrm{d}x. \end{aligned} \tag{7.4.126}$$

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



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

$$\mathsf{P}\{S-X\leq u\} = 0, \quad \text{for } u\leq -10. \tag{7.4.127}$$

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

$$70\,\mathsf{P}\{S - X \le u\} = \frac{(10+u)^2}{2}, \quad \text{for } -10 \le u \le -3, \tag{7.4.128}$$

where we multiply with 70 to get the normalization right.

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

$$70\,\mathsf{P}\{S - X \le u\} = (3+u)7 + \frac{(10-3)^2}{2} = 7u + \frac{91}{2}, \quad \text{for } -3 \le u \le 0. \tag{7.4.129}$$

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 parallelogram plus the triangle below the line $s = x - 3$. The area is

$$70\,\mathsf{P}\{S - X \le u\} = \frac{7^2}{2} - \frac{(7-u)^2}{2} + 3\cdot 7 + \frac{49}{2} = 7u - \frac{u^2}{2} + \frac{91}{2}, \quad \text{for } 0 \le u \le 7. \tag{7.4.130}$$

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

$$70\,\mathsf{P}\{S - X \le u\} = 70, \quad \text{for } 7 \le u. \tag{7.4.131}$$

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

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

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

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

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

s.2.3.18. Here is my solution in python.

```
>>> import numpy as np

>>> m = 3
>>> N = 10

>>> one = np.ones(m, dtype=int)  #  vector with ones

>>> X = np.ones(N + 1, dtype=int)
>>> S = 5 * np.ones(N, dtype=int)
>>> w = np.zeros(m, dtype=int)
>>> W = J = A = D = 0

>>> for k in range(1, N):
...     s = w.argmin()  # server chosen
```

```
...     W = w[s]  # waiting time
...     J = W + S[k]  # sojourn time
...     A += X[k]  # arrival time
...     D = A + J  # departure time
...     print(k, S[k], W, w)
...     # now update w
...     w[s] += S[k]
...     w = np.maximum(0, w - X[k + 1] * one)
...
1 5 0 [0 0 0]
2 5 0 [4 0 0]
3 5 0 [3 4 0]
4 5 2 [2 3 4]
5 5 2 [6 2 3]
6 5 2 [5 6 2]
7 5 4 [4 5 6]
8 5 4 [8 4 5]
9 5 4 [7 8 4]
>>> #
```

s.2.4.1. Let $I_j = \int_0^\infty e^{-x} x^j \, dx$. Then, $I_j = j!$, since by recursion,

$$I_j = \int_0^\infty e^{-x} x^j \, dx = -e^{-x} x^j \Big|_0^\infty + j \int_0^\infty e^{-x} x^{j-1} \, dx = j I_{j-1}, \tag{7.4.132}$$

$$I_0 = \int_0^\infty e^{-x} \, dx = 1. \tag{7.4.133}$$

By the change of variable $\lambda x \to y$, $\int_0^\infty e^{-\lambda x} x^j \, dx = I_j/\lambda^{j+1} = j!/\lambda^{j+1}$. Hence, $\mathsf{E}[X] = \lambda\, 1!/\lambda^2$.

s.2.4.2. $\mathsf{E}\left[X^2\right] = \int_0^\infty t^2 \lambda e^{-\lambda t} \, dt = \lambda\, 2!/\lambda^3$.

s.2.4.3. $\mathsf{V}[X] = \mathsf{E}\left[X^2\right] - (\mathsf{E}[X])^2 = \frac{2}{\lambda^2} - (\frac{1}{\lambda})^2 = \lambda^{-2}$.

s.2.4.4.

$$M_X(t) = \mathsf{E}[\exp(tX)] = \int_0^\infty e^{tx} f(x) \, dx = \int_0^\infty e^{tx} \lambda e^{-\lambda x} \, dx = \frac{\lambda}{\lambda - t}. \tag{7.4.134}$$

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

s.2.4.5. $M_X'(t) = \lambda/(\lambda - t)^2 \implies M_X'(0) = 1/\lambda$, $M_X''(t) = 2\lambda/(\lambda - t)^3$

s.2.4.6. $C^2 = \mathsf{V}[X]/(\mathsf{E}[X])^2 = 1/\lambda^2/(1/\lambda^2)$.

s.2.4.7. By the definition of conditional probability

$$\begin{aligned} \mathsf{P}\{X > t + h \,|\, X > t\} &= \frac{\mathsf{P}\{X > t + h, X > t\}}{\mathsf{P}\{X > t\}} = \frac{\mathsf{P}\{X > t + h\}}{\mathsf{P}\{X > t\}} \\ &= \frac{e^{-\lambda(t+h)}}{e^{-\lambda t}} = e^{-\lambda h} = \mathsf{P}\{X > h\}. \end{aligned} \tag{7.4.135}$$

s.2.4.8. $\mathsf{P}\{X_1 > t\} = \mathsf{P}\{N(t) = 0\} = e^{-\lambda t}$.

s.2.4.9. $\mathsf{E}[A_i] = \mathsf{E}\left[\sum_{k=1}^{i} X_k\right] = i\,\mathsf{E}[X] = i/\lambda$, as X_i i.i.d.

s.2.4.10. Using the i.i.d. property of the $\{X_i\}$,

$$M_{A_i}(t) = \mathsf{E}\left[e^{tA_i}\right] = \mathsf{E}\left[\exp\left(t\sum_{k=1}^{i} X_k\right)\right] = \prod_{k=1}^{i} \mathsf{E}\left[e^{tX_k}\right] = \left(\frac{\lambda}{\lambda - t}\right)^i. \tag{7.4.136}$$

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

s.2.4.11.

$$\mathsf{E}[A_i] = \int_0^\infty t f_{A_i}(t)\,\mathrm{d}t = \int_0^\infty t\lambda e^{-\lambda t}\frac{(\lambda t)^{i-1}}{(i-1)!}\,\mathrm{d}t = \frac{1}{(i-1)!}\int_0^\infty e^{-\lambda t}(\lambda t)^i\,\mathrm{d}t = \frac{i!}{(i-1)!\lambda} \tag{7.4.137}$$

As a check,

$$\mathsf{E}[A_i] = \frac{\mathrm{d}}{\mathrm{d}t}M_{A_i}(t)\bigg|_{t=0} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\lambda}{\lambda-t}\right)^i\bigg|_{t=0} = i\left(\frac{\lambda}{\lambda-t}\right)^{i-1}\frac{\lambda}{(\lambda-t)^2}\bigg|_{t=0} \tag{7.4.138}$$

s.2.4.12. With the density of A_{k+1} and applying partial integration,

$$\mathsf{P}\{A_{k+1} \le t\} = \lambda\int_0^t \frac{(\lambda s)^k}{k!}e^{-\lambda s}\,\mathrm{d}s = \lambda\frac{(\lambda s)^k}{k!}\frac{e^{-\lambda s}}{-\lambda}\bigg|_0^t + \lambda\int_0^t \frac{(\lambda s)^{k-1}}{(k-1)!}e^{-\lambda s}\,\mathrm{d}s \tag{7.4.139}$$

$$= -\frac{(\lambda t)^k}{k!}e^{-\lambda t} + \mathsf{P}\{A_k \le t\}. \tag{7.4.140}$$

s.2.4.13. $\mathsf{P}\{Z > x\} = \mathsf{P}\{\min\{X,S\} > x\} = \mathsf{P}\{X > x, S > x\} = \mathsf{P}\{X > x\}\,\mathsf{P}\{S > x\} = e^{-\lambda x}e^{-\mu x}$, as X and S independent.

s.2.4.14.

$$\mathsf{P}\{X \le S\} = \mathsf{E}[\mathbb{1}_{X\le S}] = \int_0^\infty\int_0^\infty \mathbb{1}_{x\le y} f_{X,S}(x,y)\,\mathrm{d}y\,\mathrm{d}x \tag{7.4.141}$$

$$= \lambda\mu\int_0^\infty\int_0^\infty \mathbb{1}_{x\le y}e^{-\lambda x}e^{-\mu y}\,\mathrm{d}y\,\mathrm{d}x = \lambda\mu\int_0^\infty e^{-\mu y}\int_0^y e^{-\lambda x}\,\mathrm{d}x\,\mathrm{d}y \tag{7.4.142}$$

$$= \mu\int_0^\infty e^{-\mu y}(1 - e^{-\lambda y})\,\mathrm{d}y = \mu\int_0^\infty (e^{-\mu y} - e^{-(\lambda+\mu)y})\,\mathrm{d}y \tag{7.4.143}$$

$$= 1 - \frac{\mu}{\lambda+\mu} = \frac{\lambda}{\lambda+\mu}. \tag{7.4.144}$$

With [2.2.13] it's immediate.

s.3.2.1. From the recursion and the hints,

$$\begin{aligned} L_k - Z_k &= \max\{\max\{L_{k-2} - Z_{k-2}, -Z_{k-1}\}, -Z_k\} \\ &= \max\{L_{k-2} - Z_{k-2}, -Z_{k-1}, -Z_k\} \\ &= \max\{L_0 - Z_0, -Z_1, \ldots, -Z_k\} \\ &= \max\{0, -Z_1, \ldots, -Z_k\} \\ &= -\min\{0, Z_1, \ldots, Z_k\}. \end{aligned} \tag{7.4.145}$$

s.3.2.2.

$$\begin{aligned}
\mathsf{P}\{Z_k = n\} &= \mathsf{P}\{m + N_{\lambda k} - N_{\mu k} = n\} = \mathsf{P}\{N_{\lambda k} - N_{\mu k} = n - m\} && (7.4.146)\\
&= \sum_{j=0}^{\infty} \mathsf{P}\{N_{\lambda k} = n - m + j, N_{\mu k} = j\} = \sum_{j=0}^{\infty} e^{-\lambda k}\frac{(\lambda k)^{n-m+j}}{(n-m+j)!} e^{-\mu k}\frac{(\mu k)^j}{j!} && (7.4.147)\\
&= e^{-(\lambda+\mu)k}(\lambda k)^{n-m}\sum_{j=0}^{\infty}\frac{(\lambda\mu k^2)^j}{j!(n-m+j)!}. && (7.4.148)
\end{aligned}$$

s.3.2.3. Under conditions you can find on the internet, $(W_n - \mu_n)/\sigma_n \to \mathcal{N}(0,1)$ as $n \to \infty$, where $\mathcal{N}(0,1)$ is a normally distributed random variable with $\mu = 0$ and $\sigma^2 = 1$. But then

$$\frac{W_n - \mu_n}{\sigma_n} \approx \mathcal{N}(0,1) \iff W_n - \mu_n \approx \sigma_n \mathcal{N}(0,1) \iff W_n - \mu_n \approx \mathcal{N}(0,\sigma_n^2) \iff \quad (7.4.149)$$

$$W_n \approx \mu_n + \mathcal{N}(0,\sigma_n^2) \iff W_n \approx \mathcal{N}(\mu_n,\sigma_n^2) = \mathcal{N}(n\,\mathsf{E}[S], n\,\mathsf{V}[S]), \quad (7.4.150)$$

s.3.3.1. If $A(t) = 3t^2$, then clearly $A(t)/t = 3t$. This does not converge to a limit.

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

$$\lambda(t) = \begin{cases} 1 & \text{if } 2^{2k} \leq t < 2^{2k+1} \\ 0 & \text{if } 2^{2k+1} \leq t < 2^{2(k+1)}, \end{cases} \quad (7.4.151)$$

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

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

s.3.3.2. Since $L(t) = L(0) + A(t) - D(t)$,

$$\lambda = \lim_{t\to\infty}\frac{A(t)}{t} = \lim_{t\to\infty}\frac{D(t)+L(t)}{t} = \lim_{t\to\infty}\frac{D(t)}{t} + \lim_{t\to\infty}\frac{L(t)}{t} = \delta. \quad (7.4.152)$$

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

s.3.3.3. $0 >= \mathsf{E}[S_{k-1} - X_k] = \mathsf{E}[S_{k-1}] - \mathsf{E}[X_k] = \mathsf{E}[S] - \mathsf{E}[X]$, where we use the fact that the $\{S_k\}$ and $\{X_k\}$ are i.i.d. sequences. Hence,

$$\mathsf{E}[X] > \mathsf{E}[S] \iff \frac{1}{\mathsf{E}[S]} > \frac{1}{\mathsf{E}[X]} \iff \mu > \lambda. \quad (7.4.153)$$

s.3.3.4. 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 utilization 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.3.4.1. Take $L(0) = 0$, $X_k = 10$ and $S_k = 10 - \epsilon$ for some tiny $\epsilon > 0$. Then $L(t) = 1$ nearly all of the time. In fact, $\lim_{t\to\infty} t^{-1}\int_0^t L(t)\,\mathrm{d}t = 1 - \epsilon/10$. However, $L(A_k-) = 0$ for all k.

s.3.4.2.

$$\mathsf{E}[L] = \lim_{t\to\infty} \frac{1}{t}\int_0^t L(s)\,\mathrm{d}s \neq \lim_{t\to\infty} \frac{L(t)}{t}. \tag{7.4.154}$$

If $L(t) = 1$ for all t, $\mathsf{E}[L] = 1$, but $L(t)/t \to 0$.

s.3.4.3. The idea is like this. The dictionary `L_count` counts the number of jobs that see 0, 1, and so on, in the system.

Here is the code. As an aside, it was hard to make it this simple.

```
>>> from collections import defaultdict

>>> L_count = defaultdict(int)

>>> a = [0, 2, 5, 1, 2]
>>> c = [0, 1, 1, 1, 1]

>>> d = [0] * len(a)
>>> L = [0] * len(a)

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

>>> # normalize
>>> tot = sum(L_count.values())
>>> L_dist = {k: v / tot for k, v in L_count.items()}

>>> print(L_count)
defaultdict(<class 'int'>, {0: 1, 1: 2, 2: 1, 3: 1, 4: 1, 5: 3, 6: 1})
>>> print(L_dist)
{0: 0.1, 1: 0.2, 2: 0.1, 3: 0.1, 4: 0.1, 5: 0.3, 6: 0.1}
```

s.4.1.1. When processing times at a station are nearly constant, and the jobs of this station are sent to a second station for further processing, the inter-arrival times at the second station must be roughly equal. But then the inter-arrival times are not well approximated by the exponential distribution, consequently, the arrival process is not well described by a Poisson process.

s.4.1.2. $\rho = \lambda\,\mathsf{E}[S] = 1/60 \cdot 50 = 5/6$. 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 inter-arrival times, and also not in the service times, thus $C_a^2 = C_s^2 = 0$. Also $\mathsf{E}[W] = 0$ since $\mathsf{E}[Q] = 0$.

s.4.1.3. 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.

$$\begin{aligned}
\mathsf{E}[S] &= \frac{20}{2} + \frac{80}{2} = 50 \\
\mathsf{E}\left[S^2\right] &= \frac{400}{2} + \frac{6400}{2} = 3400 \\
\mathsf{V}[S] &= \mathsf{E}[S]^2 - (\mathsf{E}[S])^2 = 3400 - 2500 = 900 \\
C_s^2 &= \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} = \frac{900}{2500} = \frac{9}{25}.
\end{aligned} \tag{7.4.155}$$

s.4.1.4. First the $G/G/1$ case. Observe that in this case, the inter-arrival time $X \sim U[3,9]$, that is, never smaller than 3 minutes, and never longer than 9 minutes.

```
>>> a = 3.0
>>> b = 9.0
>>> EX = (b + a) / 2.0  # expected inter-arrival time
>>> EX
6.0
>>> labda = 1.0 / EX  # per minute
>>> labda
0.16666666666666666
>>> VA = (b - a) * (b - a) / 12.0
>>> CA2 = VA / (EX * EX)
>>> CA2
0.08333333333333333

>>> ES = 5.0
>>> sigma = 2
>>> VS = sigma * sigma
>>> CS2 = VS / (ES * ES)
>>> CS2
0.16

>>> rho = labda * ES
>>> rho
0.8333333333333333

>>> W = (CA2 + CS2) / 2.0 * rho / (1.0 - rho) * ES
>>> W
3.041666666666665
```

Now the $M/G/1$ case. In that case $C_a^2 = 1$.

```
>>> W = (1.0 + CS2) / 2.0 * rho / (1.0 - rho) * ES
>>> W
14.499999999999993
```

The arrival process with uniform inter-arrival times is much more regular than a Poisson process. In the first case, bus arrivals are spaced in time at least with 3 minutes.

s.4.1.5. Observe that $\lambda(1-\beta)$ is the net arrival rate, as jobs are lost at a rate $\lambda\beta$. The rate at which the station carries out work is $\mu\,\mathsf{E}[L_s]$. Since all jobs that enter the system, must also leave the system (recall, the queue is finite), it follows that $\mu\,\mathsf{E}[L_s]=\lambda(1-\beta)$, from which the expression for β readily follows.

s.9. With the hint,

$$\begin{aligned}
\mathsf{E}[S] &= \mathsf{E}[\mathbb{1}_{T=1}S_1]+\mathsf{E}[\mathbb{1}_{T=2}S_2] && (7.4.156)\\
&= \mathsf{E}[\mathbb{1}_{T=1}]\,\mathsf{E}[S_1]+\mathsf{E}[\mathbb{1}_{T=2}]\,\mathsf{E}[S_2], \quad \text{by the independence of } T, && (7.4.157)\\
&= \mathsf{P}\{T=1\}\,\mathsf{E}[S_1]+\mathsf{P}\{T=2\}\,\mathsf{E}[S_2] && (7.4.158)\\
&= p\,\mathsf{E}[S_1]+q\,\mathsf{E}[S_2]. && (7.4.159)
\end{aligned}$$

For the variance, we need some algebra. Since,

$$\mathbb{1}_{T=1}\mathbb{1}_{T=2}=0 \text{ and } \mathbb{1}_{T=1}^2=\mathbb{1}_{T=1}, \tag{7.4.160}$$

we get

$$\begin{aligned}
\mathsf{V}[S] &= \mathsf{E}\left[S^2\right]-(\mathsf{E}[S])^2 && (7.4.161)\\
&= \mathsf{E}\left[(\mathbb{1}_{T=1}S_1+\mathbb{1}_{T=2}S_2)^2\right]-(\mathsf{E}[S])^2 && (7.4.162)\\
&= \mathsf{E}\left[\mathbb{1}_{T=1}S_1^2+\mathbb{1}_{T=2}S_2^2\right]-(\mathsf{E}[S])^2 && (7.4.163)\\
&= p\,\mathsf{E}\left[S_1^2\right]+q\,\mathsf{E}\left[S_2^2\right]-(\mathsf{E}[S])^2 && (7.4.164)\\
&= p\,\mathsf{V}[S_1]+p(\mathsf{E}[S_1])^2+q\,\mathsf{V}[S_2]+q(\mathsf{E}[S_2])^2-(\mathsf{E}[S])^2 && (7.4.165)\\
&= p\,\mathsf{V}[S_1]+p(\mathsf{E}[S_1])^2+q\,\mathsf{V}[S_2]+q(\mathsf{E}[S_2])^2-p^2(\mathsf{E}[S_1])^2-q^2(\mathsf{E}[S_2])^2-2pq\,\mathsf{E}[S_1]\,\mathsf{E}[S_2] && (7.4.166)\\
&= p\,\mathsf{V}[S_1]+q\,\mathsf{V}[S_2]+pq(\mathsf{E}[S_1])^2+pq(\mathsf{E}[S_2])^2-2pq\,\mathsf{E}[S_1]\,\mathsf{E}[S_2], \quad \text{as } p=1-q && (7.4.167)\\
&= p\,\mathsf{V}[S_1]+q\,\mathsf{V}[S_2]+pq(\mathsf{E}[S_1]-\mathsf{E}[S_2])^2. && (7.4.168)
\end{aligned}$$

s.4.2.1. First check the load.

```
>>> labda = 3  # per hour
>>> ES0 = 15.0 / 60  # hour
>>> ES0
0.25
>>> ER = 2.0
>>> Bmin = labda * ER / (1 - labda * ES0)
>>> Bmin
24.0

>>> B = 30
>>> ES = ES0 + ER / B
>>> rho = labda * ES
>>> rho
0.95
```

The time to form a red batch is

```
>>> labda_r = 0.5
>>> EW_r = (B - 1) / (2 * labda_r)
>>> EW_r  # in hours
29.0
```

And the time to form a blue batch is

```
>>> labda_b = labda - labda_r
>>> EW_b = (B - 1) / (2 * labda_b)
>>> EW_b  # in hours
5.8
```

The time a batch spends in queue.

```
>>> Cae = 1.0
>>> CaB = Cae / B
>>> CaB
0.03333333333333333
>>> Ce = 1.0  # SCV of service times
>>> VS0 = Ce * ES0 * ES0
>>> VS0
0.0625
>>> VR = 1.0 * 1.0  # Var setups is sigma squared
>>> VS = B * VS0 + VR
>>> VS
2.875
>>> ESb = B * ES0 + ER
>>> ESb
9.5
>>> CeB = VS / (ESb * ESb)
>>> CeB
0.03185595567867036
>>> EW = (CaB + CeB) / 2 * rho / (1 - rho) * ESb
>>> EW
5.8833333333333275
```

The time to unpack the batch, i.e., the time at the server.

```
>>> Eunpack = ER + (B - 1) / 2 * ES0 + ES0
>>> Eunpack
5.875
```

The overall time red jobs spend in the system.

```
>>> total = EW_r + EW + Eunpack
>>> total
40.758333333333326
```

s.4.2.2. Suppose a batch is just finished. The first job of a new batch needs to wait, on average, $B-1$ inter-arrival times until the batch is complete, the second $B-2$ inter-arrival times, and so on. The last job does not have to wait at all. Thus, the total time to form a batch is $(B-1)/\lambda_r$. An arbitrary job can be anywhere in the batch, hence its expected time is half the total time.

s.4.2.3. The variance of the inter-arrival time of batches is B times the variance of job inter-arrival times. The inter-arrival times of batches is also B times the inter-arrival times of jobs. Thus,

$$C_{a,B}^2 = \frac{B\,\mathsf{V}[X]}{(B\,\mathsf{E}[X])^2} = \frac{\mathsf{V}[X]}{(\mathsf{E}[X])^2}\frac{1}{B} = \frac{C_a^2}{B}. \tag{7.4.169}$$

s.4.2.4. The variance of a batch is $\mathsf{V}\left[\sum_{i=1}^B S_{0,i} + R\right] = B\,\mathsf{V}[S_0] + \mathsf{V}[R]$, since the normal service times $S_{0,i}, i = 1,\dots,B$, of the jobs are independent, and also independent of the setup time R of the batch.

s.4.2.5. First, wait until the setup is finished, then wait (on average) for half of the batch (minus the job itself) to be served, and then the job has to be served itself, that is, $\mathsf{E}[R] + \frac{B-1}{2}\mathsf{E}[S_0] + \mathsf{E}[S_0]$.

s.4.3.1. First we determine the load.

```
>>> EB = 30
>>> p = 1 / EB
>>> ES0 = 1.5
>>> labda = 9.0 / (2 * 8)  # arrival rate per hour
>>> ER = 5.0
>>> ES = ES0 + p * ER
>>> ES
1.6666666666666667
>>> rho = labda * ES
>>> rho
0.9375
```

So, at least the system is stable.

```
>>> VS0 = 0.5 * 0.5
>>> VR = 2.0 * 2.0
>>> VS = VS0 + p * VR + p * (1 - p) * ER * ER
>>> VS
1.1888888888888887
>>> Ce2 = VS / (ES * ES)
>>> Ce2
0.42799999999999999
```

And now we can fill in the waiting time formula.

```
>>> Ca2 = 1  # Poisson arrivals
>>> EW = (Ca2 + Ce2) / 2 * rho / (1 - rho) * ES
>>> EW
17.849999999999998
```

```
>>> EJ = EW + ES
>>> EJ
19.516666666666666
```

s.4.3.2. We can use the model of Section 4.2.

```
>>> B = 20
>>> ER = 4.5
>>> VR = 0
>>> ES = ES0 + ER / B
>>> ES
1.725
>>> rho = labda * ES
>>> rho
0.9703125
>>> VS = VS0 + VR / B
>>> Ce2 = VS / (ES * ES)
>>> Ce2
0.08401596303297626
>>> EW = (Ca2 + Ce2) / 2 * rho / (1 - rho) * ES
>>> EW
30.55855263157897
>>> EJ = EW + ES
>>> EJ
32.28355263157897
```

Comparing this to the results of [4.3.1], we see that the load becomes somewhat higher. Since ρ becomes close to one, doing adjustments regularly is not a good idea.

s.4.3.3.

$$\mathsf{E}[S] = (1-p)\mathsf{E}[S_0] + p(\mathsf{E}[R] + \mathsf{E}[S_0]) = \mathsf{E}[S_0] + p\,\mathsf{E}[R]. \tag{7.4.170}$$

s.4.3.4.

$$\begin{aligned} \mathsf{E}\left[S^2\right] &= (1-p)\mathsf{E}\left[S_0^2\right] + p\,\mathsf{E}\left[(S_0+R)^2\right] && (7.4.171)\\ &= (1-p)\mathsf{E}\left[S_0^2\right] + p\,\mathsf{E}\left[S_0^2\right] + 2p\,\mathsf{E}[S_0]\,\mathsf{E}[R] + p\,\mathsf{E}\left[R^2\right] && (7.4.172)\\ &= \mathsf{E}\left[S_0^2\right] + 2p\,\mathsf{E}[S_0]\,\mathsf{E}[R] + p\,\mathsf{E}\left[R^2\right]. && (7.4.173) \end{aligned}$$

s.4.3.5.

$$\begin{aligned} \mathsf{V}[S] &= \mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2 \\ &= \mathsf{E}\left[S_0^2\right] + 2p\,\mathsf{E}[S_0]\,\mathsf{E}[R] + p\,\mathsf{E}\left[R^2\right] \\ &\quad - (\mathsf{E}[S_0])^2 - 2p\,\mathsf{E}[S_0]\,\mathsf{E}[R] - (p\,\mathsf{E}[R])^2 \\ &= \mathsf{V}[S_0] + p(\mathsf{E}\left[R^2\right] - (\mathsf{E}[R])^2) + (\mathsf{E}[R])^2 p(1-p) \\ &= \mathsf{V}[S_0] + p\,\mathsf{V}[R] + (\mathsf{E}[R])^2 p(1-p) \\ &= \mathsf{V}[S_0] + p\,\mathsf{V}[R] + p^3(\mathsf{E}[R])^2 \frac{1-p}{p^2} \\ &= \mathsf{V}[S_0] + p\,\mathsf{V}[R] + p^3(\mathsf{E}[R])^2\,\mathsf{V}[B] \\ &= \mathsf{V}[S_0] + p\,\mathsf{V}[R] + p(\mathsf{E}[R])^2 C_B^2. \end{aligned} \tag{7.4.174}$$

s.4.4.1. Let's first check that $\rho < 1$.

```
>>> labda = 4.0
>>> ES0 = 10.0 / 60  # in hours
>>> labda_f = 1.0 / 3
>>> ER = 30.0 / 60  # in hours
>>> A = 1.0 / (1 + labda_f * ER)
>>> A
0.8571428571428571
>>> ES = ES0 / A
>>> ES
0.19444444444444445
>>> rho = labda * ES
>>> rho
0.7777777777777778
```

```
>>> Ca2 = 1.0
>>> C02 = 0.0  # deterministic service times
>>> Ce2 = C02 + 2 * A * (1 - A) * ER / ES0
>>> Ce2
0.7346938775510207
>>> EW = (Ca2 + Ce2) / 2 * rho / (1 - rho) * ES
>>> EW
0.5902777777777779
>>> EW + ES  # = EJ
0.7847222222222223
```

s.4.4.2. The time to fail is the time in between two interruptions. By assumption, the failure times are $\text{Exp}(\lambda_f)$, hence $m_f = 1/\lambda_f$. The expected duration of an interruption is $\mathsf{E}[R]$. With this

$$A = \frac{m_f}{m_f + \mathsf{E}[R]} = \frac{1/\lambda_f}{1/\lambda_f + \mathsf{E}[R]}. \tag{7.4.175}$$

s.4.4.3. The expectation of the *fixed* sum of random variables is the sum of the expectations, hence, as the $\{R_i\}$ are i.i.d., $\mathsf{E}\left[\sum_{i=1}^n R_i\right] = n\,\mathsf{E}[R]$.

s.4.4.4.

$$\mathsf{E}\left[\sum_{i=1}^N R_i\right] = \mathsf{E}\left[\sum_{n=0}^\infty \mathbb{1}_{N=n}\left(\sum_{i=1}^n R_i\right)\right] = \sum_{n=0}^\infty \mathsf{E}[\mathbb{1}_{N=n} n\,\mathsf{E}[R]] \tag{7.4.176}$$

$$= \mathsf{E}[R]\sum_{n=0}^\infty n\,\mathsf{E}[\mathbb{1}_{N=n}] = \mathsf{E}[R]\sum_{n=0}^\infty n p_n = \mathsf{E}[R]\,\mathsf{E}[N]. \tag{7.4.177}$$

s.4.4.5. If $S_0 = s$, then the expected number of failures that arrive is $\sim P(\lambda_f s)$. Therefore, $\mathsf{E}[N] = \mathsf{E}\left[\lambda_f S_0\right] = \lambda_f\,\mathsf{E}[S_0]$.

In more detail, and with the hint,

$$\mathsf{E}[N] = \int_0^\infty \sum_{k=0}^\infty k e^{-\lambda_f s}\frac{(\lambda_f s)^k}{k!} g(s)\,\mathrm{d}s = \int_0^\infty \lambda_f s g(s)\,\mathrm{d}s = \lambda_f\,\mathsf{E}[S_0]. \tag{7.4.178}$$

Next, [2.2.8],

$$\mathsf{E}\left[N^2\right] = \int_0^\infty \sum_{k=0}^\infty k^2 e^{-\lambda_f s} \frac{(\lambda_f s)^k}{k!} g(s)\,\mathrm{d}s = \int_0^\infty (\lambda_f^2 s^2 + \lambda_f s) g(s)\,\mathrm{d}s. \tag{7.4.179}$$

s.4.4.6.

$$\mathsf{E}[S] = \mathsf{E}[S_0] + \mathsf{E}[N]\,\mathsf{E}[R] = \mathsf{E}[S_0] + \lambda_f\,\mathsf{E}[S_0]\,\mathsf{E}[R] = \mathsf{E}[S_0](1 + \lambda_f\,\mathsf{E}[R]). \tag{7.4.180}$$

s.4.4.7. Observe that S_0 and N are not independent. In fact, when $S_0 = s$, the number of failures N is Poisson distributed with mean $\lambda_f s$.

s.4.4.8. Just work out the square of $S_0 + \sum_{i=1}^N R_i$ and take expectations. Realize that $(\sum_i R_i)^2 = \sum_i R_i^2 + \sum_i \sum_{j\neq i} R_i R_j$.

s.4.4.9.

$$\mathsf{E}\left[S_0 \sum_{i=1}^N R_i\right] = \mathsf{E}\left[S_0 \sum_{n=0}^\infty \mathbb{1}_{N=n} \sum_{i=1}^N R_i\right] = \mathsf{E}\left[S_0 \sum_{n=0}^\infty \mathbb{1}_{N=n} \sum_{i=1}^n R_i\right] \tag{7.4.181}$$

$$= \mathsf{E}[R]\,\mathsf{E}\left[S_0 \sum_{n=0}^\infty \mathbb{1}_{N=n} n\right] = \mathsf{E}[R] \sum_{n=0}^\infty n\,\mathsf{E}[S_0\,\mathbb{1}_{N=n}]. \tag{7.4.182}$$

Now observe that S_0 and $\mathbb{1}_{N=n}$ are not independent. Thus, we need to use the joint distribution of [4.4.5], and then,

$$\mathsf{E}[S_0\,\mathbb{1}_{N=n}] = \int_0^\infty s g(s) e^{-\lambda_f s} \frac{(\lambda_f s)^n}{n!}\,\mathrm{d}s. \tag{7.4.183}$$

Continuing with the previous relation:

$$\mathsf{E}[R] \sum_{n=0}^\infty n\,\mathsf{E}[S_0\,\mathbb{1}_{N=n}] = \mathsf{E}[R] \sum_{n=0}^\infty n \int_0^\infty s g(s) e^{-\lambda_f s} \frac{(\lambda_f s)^n}{n!}\,\mathrm{d}s = \mathsf{E}[R] \int_0^\infty s g(s) \sum_{n=0}^\infty n e^{-\lambda_f s} \frac{(\lambda_f s)^n}{n!}\,\mathrm{d}s \tag{7.4.184}$$

$$= \mathsf{E}[R] \int_0^\infty s g(s) \lambda_f s\,\mathrm{d}s = \lambda_f\,\mathsf{E}[R]\,\mathsf{E}\left[S_0^2\right]. \tag{7.4.185}$$

s.4.4.10. With the hint,

$$\mathsf{E}\left[\sum_{i=1}^N R_i^2\right] = \mathsf{E}\left[R^2\right]\mathsf{E}[N] = \lambda_f\,\mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right]. \tag{7.4.186}$$

s.4.4.11. Since the $\{R_i\}$ are i.i.d.,

$$\mathsf{E}\left[\sum_{i=1}^N \sum_{j\neq i} R_i R_j\right] = \mathsf{E}\left[\sum_{n=0}^\infty \mathbb{1}_{N=n} \sum_{i=1}^N \sum_{j\neq i} R_i R_j\right] = \mathsf{E}\left[\sum_{n=0}^\infty \mathbb{1}_{N=n} \sum_{i=1}^n \sum_{j\neq i} R_i R_j\right] \tag{7.4.187}$$

$$= \mathsf{E}\left[\sum_{n=0}^\infty \mathbb{1}_{N=n} n(n-1)\right] (\mathsf{E}[R])^2 = (\mathsf{E}\left[N^2\right] - \mathsf{E}[N])(\mathsf{E}[R])^2 \tag{7.4.188}$$

$$= (\lambda_f^2\,\mathsf{E}\left[S_0^2\right] + \lambda_f\,\mathsf{E}[S_0] - \lambda\,\mathsf{E}[S_0])(\mathsf{E}[R])^2. \tag{7.4.189}$$

where we use [4.4.5] in the last line.

s.4.4.12. It is just algebra based on the results above. Observe that $(1/A) = 1 + \lambda_f \mathsf{E}[R]$,

$$\begin{aligned}
\mathsf{E}\left[S^2\right] &= \mathsf{E}\left[S_0^2\right] + 2\,\mathsf{E}\left[S_0 \sum_{i=1}^N R_i\right] + \mathsf{E}\left[\sum_{i=1}^N R_i^2\right] + \mathsf{E}\left[\sum_{i=1}^N \sum_{j\neq i} R_i R_j\right] && (7.4.190)\\
&= \mathsf{E}\left[S_0^2\right] + 2\lambda_f \mathsf{E}[R]\,\mathsf{E}\left[S_0^2\right] + \lambda_f \mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right] + \lambda_f^2 \mathsf{E}\left[S_0^2\right](\mathsf{E}[R])^2 && (7.4.191)\\
&= \mathsf{E}\left[S_0^2\right]/A + \lambda_f \mathsf{E}[R]\,\mathsf{E}\left[S_0^2\right] + \lambda_f \mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right] + \lambda_f^2 \mathsf{E}\left[S_0^2\right](\mathsf{E}[R])^2 && (7.4.192)\\
&= \mathsf{E}\left[S_0^2\right]/A + \lambda_f \mathsf{E}[R]\,\mathsf{E}\left[S_0^2\right](1 + \lambda_f \mathsf{E}[R]) + \lambda_f \mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right] && (7.4.193)\\
&= \mathsf{E}\left[S_0^2\right]/A + \lambda_f \mathsf{E}[R]\,\mathsf{E}\left[S_0^2\right]/A + \lambda_f \mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right] && (7.4.194)\\
&= (1 + \lambda_f \mathsf{E}[R])\mathsf{E}\left[S_0^2\right]/A + \lambda_f \mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right] && (7.4.195)\\
&= \mathsf{E}\left[S_0^2\right]/A^2 + \lambda_f \mathsf{E}[S_0]\,\mathsf{E}\left[R^2\right]. && (7.4.196)
\end{aligned}$$

s.4.4.13.

$$\mathsf{V}[S] = \mathsf{E}\left[S^2\right] - (\mathsf{E}[S])^2 = \frac{\mathsf{E}\left[S_0^2\right]}{A^2} + \lambda_f \mathsf{E}\left[R^2\right]\mathsf{E}[S_0] - \frac{(\mathsf{E}[S_0])^2}{A^2}. \qquad (7.4.197)$$

s.4.4.14. Using [4.4.8]–[4.4.13]

$$\begin{aligned}
C_s^2 &= \frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2} = \frac{V(S)A^2}{(\mathsf{E}[S_0])^2} = \frac{\mathsf{E}\left[S_0^2\right] + \lambda_f \mathsf{E}\left[R^2\right]\mathsf{E}[S_0]A^2 - (\mathsf{E}[S_0])^2}{(\mathsf{E}[S_0])^2} && (7.4.198)\\
&= \frac{\mathsf{E}\left[S_0^2\right] - (\mathsf{E}[S_0])^2}{(\mathsf{E}[S_0])^2} + \frac{\lambda_f \mathsf{E}\left[R^2\right]\mathsf{E}[S_0]A^2}{(\mathsf{E}[S_0])^2} = C_0^2 + \frac{\lambda_f \mathsf{E}\left[R^2\right]A^2}{\mathsf{E}[S_0]}. && (7.4.199)
\end{aligned}$$

s.4.4.15. When repair times are exponentially distributed with mean $\mathsf{E}[R]$ we have, by [2.4.2], that $\mathsf{E}\left[R^2\right] = 2(\mathsf{E}[R])^2$. Since $A = 1/(1 + \lambda_f \mathsf{E}[R])$,

$$\begin{aligned}
\lambda_f \mathsf{E}\left[R^2\right]A^2 &= 2\lambda_f(\mathsf{E}[R])^2 A^2 = 2\frac{\lambda_f \mathsf{E}[R]}{1 + \lambda_f \mathsf{E}[R]} A\,\mathsf{E}[R]\\
&= 2\left(1 - \frac{1}{1 + \lambda_f \mathsf{E}[R]}\right) A\,\mathsf{E}[R] = 2(1 - A)A\,\mathsf{E}[R].
\end{aligned} \qquad (7.4.200)$$

s.4.5.1. First station 1.

```
>>> labda = 2.0
>>> S1 = 20.0 / 60
>>> rho1 = labda * S1
>>> ca1 = 2.0
>>> cs1 = 0.5
>>> EW1 = (ca1 + cs1) / 2 * rho1 / (1 - rho1) * S1
>>> EJ1 = EW1 + S1
>>> EJ1
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 = 25.0 / 60
>>> rho2 = labda * S2
>>> ca2 = cd1  # here we use our formula
>>> cs2 = 0.5
>>> EW2 = (ca2 + cs2) / 2 * rho2 / (1 - rho2) * S2
>>> EJ2 = EW2 + S2
>>> EJ2
2.3263888888888897
>>> EJ1 + EJ2
3.4930555555555562
```

s.5.1.1. It is evident that $X_k = Y(D_k) - Y(D_{k-1}) = S_k$, hence $X = \lim_{n\to\infty} n^{-1}\sum_{k=1}^n X_k = \mathsf{E}[S]$. In the relation $Y = \lambda X$, the λ is δ since we consider departure epochs $T_k = D_k$, rather than A_k. Thus, with the renewal reward theorem $Y = \lambda X$, we get that $Y = \lim_{t\to\infty} t^{-1}\int_0^t \mathbb{1}_{L(s)>0}\,\mathrm{d}s = \delta\,\mathsf{E}[S]$. Finally, by rate-stability, the job arrival rate $\lambda = \delta$, hence the long-run time-average fraction of time the server is busy is equal to $\rho = \lambda\,\mathsf{E}[S]$.

s.5.1.2. For direction of the inequalities in the equation of the hint, observe that t can lie half way a service interval, and $A(t) \geq D(t)$.

For the LHS, as $A(t)\to\infty$ as $t\to\infty$,

$$\lim_{t\to\infty}\frac{1}{t}\sum_{k=1}^{A(t)} S_k = \lim_{t\to\infty}\frac{A(t)}{t}\frac{1}{A(t)}\sum_{k=1}^{A(t)} S_k = \lim_{t\to\infty}\frac{A(t)}{t}\cdot\lim_{t\to\infty}\frac{1}{A(t)}\sum_{k=1}^{A(t)} S_k = \lambda\,\mathsf{E}[S]. \tag{7.4.201}$$

Apply similar limits to the RHS. The limit of the middle term gives the fraction of time the server is busy.

s.5.2.1.

$$A(t) = \sum_{k=1}^{\infty}\mathbb{1}_{A_k\leq t} = \sum_{k=1}^{\infty}\sum_{n=0}^{\infty}\mathbb{1}_{L(A_k-)=n}\,\mathbb{1}_{A_k\leq t} = \sum_{n=0}^{\infty} A(n,t). \tag{7.4.202}$$

s.5.2.2. If $\lambda > \delta$, then $L(t)\to\infty$. But then there must be a last time, s say, that $L(s) = n+1$, and $L(t) > n+1$ for all $t > s$. Hence, after time s no job will see the system with n jobs. Thus $A(n,t) = A(n,s)$ for all $t > s$. As this is finite, $\lim_{t\to\infty} A(n,t)/t = 0$.

s.5.2.3. It is the $D/D/1$ queue. Next, $A_k = 2(k-1)$ as jobs arrive at $t = 0,2,4,\ldots$ We also know that $L(s) = 1$ if $s\in[2k, 2k+1)$ and $L(s) = 0$ for $s\in[2k-1, 2k)$ for $k = 0,1,2,\ldots$. Thus, $L(A_k-) = L(2k-) = 0$. Hence, $A(0,t)\approx t/2$ for $t\gg 0$, and $A(n,t) = 0$ for $n\geq 1$.

s.5.2.4. Observe that the system never contains more than 1 job. Hence, $Y(n,t) = 0$ for all $n\geq 2$. Then we see that $Y(1,t) = \int_0^t \mathbb{1}_{L(s)=1}\,\mathrm{d}s$. 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 $t\in[1,2)$,

$$L(t) = 0 \implies \mathbb{1}_{L(t)=0} \implies Y(1,t) \text{ does not change.} \tag{7.4.203}$$

Continuing to $[2,3)$ and so on gives

$$Y(1,t) = \begin{cases} t & t \in [0,1), \\ 1 & t \in [1,2), \\ 1+(t-2) & t \in [2,3), \\ 2 & t \in [3,4), \\ 2+(t-4) & t \in [4,5), \end{cases} \tag{7.4.204}$$

and so on. Since $Y(n,t) = 0$ for all $n \geq 2$, $L(s) = 1$ or $L(s) = 0$ for all s, therefore,

$$Y(0,t) = t - Y(1,t). \tag{7.4.205}$$

s.5.2.5. From the other exercises:

$$\lambda(0) \approx \frac{A(0,t)}{Y(0,t)} \approx \frac{t/2}{t/2} = 1, \tag{7.4.206}$$

$$\lambda(1) \approx \frac{A(1,t)}{Y(1,t)} \approx \frac{0}{t/2} = 0, \tag{7.4.207}$$

$$p(0) \approx \frac{Y(0,t)}{t} \approx \frac{t/2}{t} = \frac{1}{2}, \tag{7.4.208}$$

$$p(1) \approx \frac{Y(1,t)}{t} \approx \frac{t/2}{t} = \frac{1}{2}. \tag{7.4.209}$$

For the rest $\lambda(n) = 0$, and $p(n) = 0$, for $n \geq 2$.

s.5.2.6. $D(0,t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \leq t, L(D_k)=0}$. From the graph of $\{L(s)\}$ we see that all jobs leave an empty system behind. Thus, $D(0,t) \approx t/2$, and $D(n,t) = 0$ for $n \geq 1$. With this, $D(0,t)/Y(1,t) \sim (t/2)/(t/2) = 1$, and so,

$$\mu(1) = \lim_{t\to\infty} \frac{D(0,t)}{Y(1,t)} = 1, \tag{7.4.210}$$

and $\mu(n) = 0$ for $n \geq 2$.

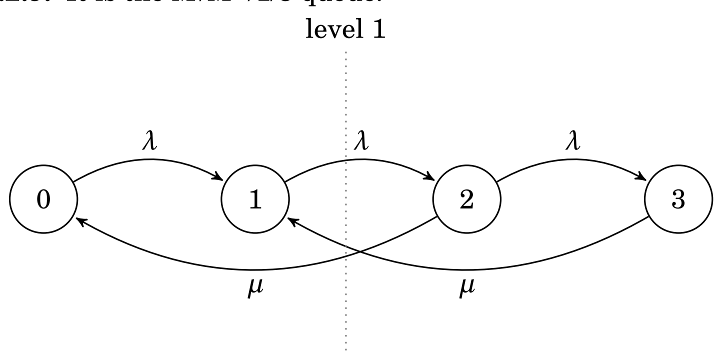
s.5.2.7. $\lambda(0)p(0) = 1 \cdot 1/2 = 1/2$, $\lambda(n)p(n) = 0$ for $n > 1$, as $\lambda(n) = 0$ for $n > 0$.
From [5.2.6], $\mu(1) = 1$, hence $\mu(1)p(1) = 1 \cdot 1/2 = 1/2$. Moreover, $\mu(n) = 0$ for $n \geq 2$.
Clearly, for all n we have $\lambda(n)p(n) = \mu(n+1)p(n+1)$.

s.5.2.8. Noting that $L(s) = \sum_{n=0}^{\infty} n \, \mathbb{1}_{L(s)=n}$,

$$\frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \frac{1}{t}\int_0^t \left(\sum_{n=0}^{\infty} n \, \mathbb{1}_{L(s)=n}\right)\mathrm{d}s = \sum_{n=0}^{\infty} \frac{n}{t}\int_0^t \mathbb{1}_{L(s)=n}\,\mathrm{d}s, \tag{7.4.211}$$

Now apply (5.2.1c).

s.5.2.9. It is the $M/M^2/1/3$ queue.



With level-crossing:

$$\lambda p(0) = \mu p(2), \quad \text{the level between 0 and 1,} \tag{7.4.212}$$
$$\lambda p(1) = \mu p(2) + \mu p(3), \quad \text{see level 1,} \tag{7.4.213}$$
$$\lambda p(2) = \mu p(3), \quad \text{the level between 2 and 3.} \tag{7.4.214}$$
$$\tag{7.4.215}$$

Solving this in terms of $p(0)$ gives $p(2) = \rho p(0)$, $p(3) = \rho p(2) = \rho^2 p(0)$, and

$$\lambda p(1) = \mu(p(2) + p(3)) = \mu(\rho + \rho^2)p(0) = (\lambda + \lambda^2/\mu)p(0), \tag{7.4.216}$$

hence $p(1) = p(0)(\mu + \lambda)/\mu$.

s.18.

$$\lim_{t\to\infty} \frac{I(n,t)}{t} = \lim_{t\to\infty} \frac{A(n-1,t)}{t} + \lim_{t\to\infty} \frac{D(n,t)}{t} = \lambda(n-1)p(n-1) + \mu(n+1)p(n+1) \tag{7.4.217}$$

and

$$\lim_{t\to\infty} \frac{O(n,t)}{t} = \lim_{t\to\infty} \frac{A(n,t)}{t} + \lim_{t\to\infty} \frac{D(n-1,t)}{t} = \lambda(n)p(n) + \mu(n)p(n) \tag{7.4.218}$$

s.5.3.1. All arrivals see an empty system. Hence, $A(0,t)/A(t) \approx (t/2)/(t/2) = 1$, and $A(n,t) = 0$ for $n > 0$. Thus, $\pi(0) = \lim_{t\to\infty} A(0,t)/A(t) = 1$ and $\pi(n) = 0$ for $n > 0$. Recall from the other exercises that $p(0) = 1/2$. Hence, statistics as obtained via time averages are not necessarily the same as statistics obtained at arrival moments (or any other point process).

s.5.3.2. From the relevant previous exercises, $\lambda = \lim_{t\to\infty} A(t)/t = 1/2$. $\lambda(0) = 1$, $p(0) = 1/2$, and $\pi(0) = 1$. Hence,

$$\lambda\pi(0) = \lambda(0)p(0) \implies \frac{1}{2} \times 1 = 1 \times \frac{1}{2}. \tag{7.4.219}$$

For $n > 0$ it's easy, everything is 0.

s.5.3.3. The assumptions lead us to conclude that $\lambda > \delta$. As a consequence, the queue length must increase in the long run (jobs come in faster than they leave). Therefore, $A(n,t)/t \to 0$ for all n, and also $D(n,t)/t \to 0$. Consequently, $\pi(n) = \delta(n) = 0$, which is the only sensible reconciliation with (5.3.8).

s.5.3.4. 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,

$$X = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} X_k = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}_{L(A_k-)=n} = \pi(n). \tag{7.4.220}$$

Since $Y = \lim_{t\to\infty} Y(t)/t = \lim_{t\to\infty} A(n,t)/t$ it follows from the renewal reward theorem that

$$Y = \lambda X \implies \lim_{t\to\infty} \frac{A(n,t)}{t} = \lambda X = \lambda\pi(n). \tag{7.4.221}$$

s.5.4.1. The dimension of λ is a *rate* and $\mathsf{E}[J]$ is time, hence the dimension of $\lambda\mathsf{E}[J]$ is a number, just like $\mathsf{E}[L]$.

s.5.4.2. Recall that $\rho := \lim_{t\to\infty} t^{-1}\int_0^t L_s(s)\,\mathrm{d}s$ for the $G/G/1$ queue.

s.5.4.3. $\mathsf{E}[W] = \mathsf{E}[J] - \mathsf{E}[S] = 1/6/(1-5/6) - 1/6 = 5/6h$.

s.5.4.4. When you arrive, you first have to wait for the job in service to finish and then for the 9 jobs in queue. By the memoryless property, at the moment you arrive, the remaining service time of the job in service is $\mathsf{E}[S]$ Thus, you have to wait $10/\mu = 5/3h \neq 5/6h$.

s.5.4.5.

$$\begin{aligned}\int_0^T L(s)\,\mathrm{d}s &= \int_0^T \sum_{k=1}^{A(T)} \mathbb{1}_{A_k\le s<D_k}\,\mathrm{d}s \\ &= \sum_{k=1}^{A(T)} \int_0^T \mathbb{1}_{A_k\le s<D_k}\,\mathrm{d}s = \sum_{k=1}^{A(T)} J_k.\end{aligned} \tag{7.4.222}$$

s.5.4.6. By the memoryless property of the (exponential) distributed service times of the $M/M/1$ queue, the duration of a job in service, if any, is $\mathrm{Exp}(\mu)$ also at an arrival moment. Therefore, at an arrival moment, all jobs in the system (whether in service or not) have the same expected duration. Hence, the expected time to spend in queue is the expected number of jobs in the system times the expected service time of each job, i.e., $\mathsf{E}[J] = \mathsf{E}[L]\,\mathsf{E}[S]$. Note that we use PASTA to see that the expected number of jobs in the system at an arrival is $\mathsf{E}[L]$. For the $M/G/1$ queue, the job in service (if any) does not have the same distribution as a job in queue. Hence, the expected time in queue is not $\mathsf{E}[L]\,\mathsf{E}[S]$.

s.5.4.7.

$$\mathsf{E}[J] = \mathsf{E}[L]\,\mathsf{E}[S] + \mathsf{E}[S] = \lambda\,\mathsf{E}[J]\,\mathsf{E}[S] + \mathsf{E}[S] = \rho\,\mathsf{E}[J] + \mathsf{E}[S], \tag{7.4.223}$$

$$\mathsf{E}[L] = \lambda\,\mathsf{E}[J] = \frac{\lambda\,\mathsf{E}[S]}{1-\rho} = \frac{\rho}{1-\rho}, \tag{7.4.224}$$

$$\mathsf{E}[W] = \mathsf{E}[J] - \mathsf{E}[S] = \frac{\mathsf{E}[S]}{1-\rho} - \mathsf{E}[S] = \frac{\rho}{1-\rho}\,\mathsf{E}[S], \tag{7.4.225}$$

$$\mathsf{E}[Q] = \lambda\,\mathsf{E}[J] = \frac{\rho^2}{1-\rho}, \tag{7.4.226}$$

$$\mathsf{E}[L_s] = \mathsf{E}[L] - \mathsf{E}[Q] = \frac{\rho}{1-\rho} - \frac{\rho^2}{1-\rho} = \rho, \tag{7.4.227}$$

s.5.4.8. Intuitively, the left term is all the work that arrived up to time t, the middle term is all the work that has been processed, and the right term all the work that left. Any job that is half-way its service counts for full at the left, for half in the middle expression, and not in the right.

More formally, for any job k and time t, we have $J_k\,\mathbb{1}_{A_k\le t} \ge \int_0^t \mathbb{1}_{A_k\le s<D_k}\,\mathrm{d}s \ge J_k\,\mathbb{1}_{D_k\le t}$. (To see this, fix k, and check the three cases $t < A_k, A_k \le t < D_k, D_k < t$.) Then,

$$\sum_{k=1}^{\infty} J_k\,\mathbb{1}_{A_k\le t} \ge \int_0^t \sum_{k=1}^{\infty} \mathbb{1}_{A_k\le t<D_k}\,\mathrm{d}s \ge \sum_{k=1}^{\infty} J_k\,\mathbb{1}_{D_k\le t}. \tag{7.4.228}$$

Finally, note that $\sum_{k=1}^{\infty} J_k\,\mathbb{1}_{A_k\le t} = \sum_{k=1}^{A(t)} J_k$ and $\sum_{k=1}^{\infty} J_k\,\mathbb{1}_{D_k\le t} = \sum_{k=1}^{D(t)} J_k$, and use the definition of $L(s)$.

s.5.4.9.

$$\lim_{t\to\infty} \frac{A(t)}{t} \frac{1}{A(t)} \sum_{k=1}^{A(t)} J_k \geq \lim_{t\to\infty} \frac{1}{t} \int_0^t L(s)\,\mathrm{d}s \geq \lim_{t\to\infty} \frac{D(t)}{t} \frac{1}{D(t)} \sum_{k=1}^{D(t)} J_k. \tag{7.4.229}$$

We use the strong law of large numbers to conclude that the limits converges to $n^{-1}\sum_{k=1}^{n} J_k \to \mathsf{E}[J]$, and we assume that $\{J_k, k \geq N\}$ is a sequence of i.i.d. random variables for N sufficiently large.

s.5.4.10. $A(t) = D(t) + 10$ for $t > 10$. However, since $\delta = \lambda$, Little's law follows by [5.4.9].

s.6.1.1.

$$M_L(s) = \mathsf{E}\left[e^{sL}\right] = \sum_{n=0}^{\infty} e^{sn} p(n) = (1-\rho)\sum_{n} e^{sn}\rho^n = \frac{1-\rho}{1-e^s\rho}, \tag{7.4.230}$$

where we assume that s is such that $e^s\rho < 1$. Then,

$$M_L'(s) = (1-\rho)\frac{1}{(1-e^s\rho)^2}e^s\rho \implies \mathsf{E}[L] = M_L'(0) = \frac{\rho}{1-\rho}, \tag{7.4.231}$$

$$\mathsf{E}\left[L^2\right] = M''(0) = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho}, \tag{7.4.232}$$

$$\mathsf{V}[L] = \mathsf{E}\left[L^2\right] - (\mathsf{E}[L])^2 = \frac{\rho(1+\rho)}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} = \frac{\rho}{(1-\rho)^2}, \tag{7.4.233}$$

$$\mathsf{P}\{L \geq n\} = \sum_{k=n}^{\infty} p(k) = \sum_{k=n}^{\infty} p(0)\rho^k = (1-\rho)\sum_{k=n}^{\infty}\rho^k = (1-\rho)\rho^n\sum_{k=0}^{\infty}\rho^k = (1-\rho)\rho^n\frac{1}{1-\rho} = \rho^n. \tag{7.4.234}$$

s.6.1.2. Use the results of [2.2.18]; use PASTA to conclude that $\beta = p(K)$.

s.6.1.3. 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., $n-1$.

$$\mathsf{E}[Q] = \sum_{n=1}^{\infty}(n-1)\pi(n) = \sum_{n=1}^{\infty} n\pi(n) - \sum_{n=1}^{\infty}\pi(n) = \mathsf{E}[L] - (1-\pi(0)) = \mathsf{E}[L] - \rho. \tag{7.4.235}$$

s.6.1.4.

$$p(n) = \frac{\lambda(n-1)}{\mu(n)}p(n-1) = \frac{\lambda}{\min\{c,n\}\mu}p(n-1) = \frac{1}{\min\{c,n\}}(c\rho)p(n-1) \tag{7.4.236}$$

$$= \frac{1}{\min\{c,n\}\min\{c,n-1\}}(c\rho)^2 p(n-2) \tag{7.4.237}$$

$$= \frac{1}{\Pi_{k=1}^{n}\min\{c,k\}}(c\rho)^n p(0). \tag{7.4.238}$$

To obtain the normalization constant G,

$$1 = \sum_{n=0}^{\infty} p(n) = \sum_{n=0}^{c-1} p(n) + \sum_{n=c}^{\infty} p(n) \tag{7.4.239}$$

$$= p(0)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + p(0)\sum_{n=c}^{\infty}\frac{c^c}{c!}\rho^n \tag{7.4.240}$$

$$= p(0)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + p(0)\sum_{n=c}^{\infty}\frac{(c\rho)^c}{c!}\rho^{n-c} \tag{7.4.241}$$

$$= p(0)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + p(0)\frac{(c\rho)^c}{c!}\sum_{n=0}^{\infty}\rho^n \tag{7.4.242}$$

$$= p(0)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + p(0)\frac{(c\rho)^c}{c!(1-\rho)}. \tag{7.4.243}$$

Next,

$$\mathsf{E}[Q] = \sum_{n=c}^{\infty}(n-c)p(n) = \sum_{n=c}^{\infty}(n-c)\frac{c^c}{c!}\rho^n p(0) \tag{7.4.244}$$

$$= \frac{c^c\rho^c}{Gc!}\sum_{n=c}^{\infty}(n-c)\rho^{n-c} = \frac{c^c\rho^c}{Gc!}\sum_{n=0}^{\infty}n\rho^n \tag{7.4.245}$$

$$= \frac{c^c\rho^c}{Gc!}\frac{\rho}{(1-\rho)^2}. \tag{7.4.246}$$

The derivation of the expected number of jobs in service becomes easier if we pre-multiply the normalization constant G:

$$G\,\mathsf{E}[L_s] = G\left(\sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n)\right) \tag{7.4.247}$$

$$= \sum_{n=1}^{c} n\frac{(c\rho)^n}{n!} + \sum_{n=c+1}^{\infty} c\frac{c^c\rho^n}{c!} = \sum_{n=1}^{c}\frac{(c\rho)^n}{(n-1)!} + \frac{c^{c+1}}{c!}\sum_{n=c+1}^{\infty}\rho^n \tag{7.4.248}$$

$$= \sum_{n=0}^{c-1}\frac{(c\rho)^{n+1}}{n!} + \frac{(c\rho)^{c+1}}{c!}\sum_{n=0}^{\infty}\rho^n = c\rho\left(\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}\right). \tag{7.4.249}$$

Observe that the RHS is precisely equal to $\rho c G$, so that $\mathsf{E}[L_s] = c\rho$.

s.6.1.5. Take $c = 1$

$$p(n) = \frac{1}{G}\frac{(c\rho)^0}{0!} = \frac{1}{G}, \quad n = 0, \dots, 1-1 \tag{7.4.250}$$

$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!} = \frac{1}{G}\frac{1^1\rho^n}{1!} = \frac{\rho^n}{G}, \quad n = 1, 1+1, \dots \tag{7.4.251}$$

$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} = \sum_{n=0}^{0}\frac{\rho^0}{0!} + \frac{\rho}{(1-\rho)} = 1 + \frac{\rho}{1-\rho} = \frac{1}{1-\rho}, \tag{7.4.252}$$

$$\mathsf{E}[L] = \frac{(c\rho)^c}{c!G}\frac{\rho}{(1-\rho)^2} = \frac{\rho}{1/(1-\rho)}\frac{\rho}{(1-\rho)^2} = \frac{\rho^2}{1-\rho}, \tag{7.4.253}$$

$$\mathsf{E}[L] = \sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) = p(1) + 1\sum_{n=2}^{\infty} p(n) = 1 - p(0) = \rho. \tag{7.4.254}$$

Everything is in accordance to the formulas we derived earlier for the $M/M/1$ queue.

s.6.1.6. In the expressions for the $M/M/c$ queue, just neglect the parts that deal with states $n > c$. We see that $p(n) = p(0)(c\rho)^n/n!$. For the normalization $1 = \sum_{n=0}^{c} p(n) = p(0)\sum_{n=0}^{c} \frac{(c\rho)^n}{n!}$. Next,

$$\mathsf{E}[L_s] = \sum_{n=0}^{c} np(n) = \sum_{n=1}^{c} np(n) \tag{7.4.255}$$

$$= G^{-1}\sum_{n=1}^{c} n\frac{(\lambda/\mu)^n}{n!} = G^{-1}\sum_{n=1}^{c}\frac{(\lambda/\mu)^n}{(n-1)!} \tag{7.4.256}$$

$$= \frac{\lambda}{\mu G}\sum_{n=0}^{c-1}\frac{(\lambda/\mu)^n}{n!} = \frac{\lambda}{G\mu}\left(G - \frac{(\lambda/\mu)^c}{c!}\right) \tag{7.4.257}$$

$$= \frac{\lambda}{\mu}\left(1 - \frac{1}{G}\frac{(\lambda/\mu)^c}{c!}\right) \tag{7.4.258}$$

$$= \frac{\lambda}{\mu}(1 - p(c)). \tag{7.4.259}$$

s.6.1.7. By taking the limit $c \to \infty$, note first that in (6.1.8b),

$$\frac{(c\rho)^c}{(1-\rho)c!} = \frac{(\lambda/\mu)^c}{(1-\rho)c!} \to 0, \quad \text{as } c \to \infty. \tag{7.4.260}$$

Hence

$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!} \to \sum_{n=0}^{\infty}\frac{(c\rho)^n}{n!} = e^{\lambda/\mu}. \tag{7.4.261}$$

Next, for any fixed n, eventually $c > n$, and then, as $\rho = \lambda/(\mu c)$,

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!} = \frac{1}{G}\frac{(\lambda/\mu)^n}{n!} \to e^{-\lambda/\mu}\frac{(\lambda/\mu)^n}{n!}, \quad \text{as } c \to \infty. \tag{7.4.262}$$

s.6.1.8.

$$p(n+1) = \frac{(N-n)\lambda}{\mu}p(n) = \rho(N-n)p(n) \tag{7.4.263}$$

$$= \rho^2(N-n)(N-(n-1))p(n-1) \tag{7.4.264}$$

$$= \rho^3(N-n)(N-(n-1))(N-(n-2))p(n-2) \tag{7.4.265}$$

$$= \rho^{n+1}(N-n)(N-(n-1))\cdots(N-(0))p(0) \tag{7.4.266}$$

$$= \rho^{n+1}\frac{N!}{(N-(n+1))!}p(0). \tag{7.4.267}$$

s.6.1.9. Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = n\mu$. Then

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)}p(n) = \frac{(N-n)\lambda}{(n+1)\mu}p(n) = \frac{(N-n)(N-(n-1))}{(n+1)n}\frac{\lambda^2}{\mu^2}p(n-1) \tag{7.4.268}$$

$$= \frac{N!}{(N-(n+1))!}\frac{1}{(n+1)!}\rho^{n+1}p(0) = \binom{N}{n+1}\rho^{n+1}p(0). \tag{7.4.269}$$

Therefore, $G = \sum_{k=0}^{N}\rho^k\binom{N}{k}$.

s.6.1.10. To take the limit $K \to \infty$—mind, not the limit $n \to \infty$—, write

$$G = \frac{1-\rho^{K+1}}{1-\rho} = \frac{1}{1-\rho} - \frac{\rho^{K+1}}{1-\rho}. \tag{7.4.270}$$

Since $\rho^{K+1} \to 0$ as $K \to \infty$ (recall, $\rho < 1$), we get

$$G \to \frac{1}{1-\rho}, \tag{7.4.271}$$

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.6.1.11.

$$\begin{aligned}
\mathsf{E}[L] &= \sum_{n=0}^{\infty} n p(n) = \sum_{n=0}^{\infty}\sum_{i=1}^{n} \mathbb{1}_{i\le n} p(n) && n = \sum_{i=1}^{n} \mathbb{1}_{i\le n} && (7.4.272)\\
&= \sum_{n=0}^{\infty}\sum_{i=1}^{\infty} \mathbb{1}_{i\le n} p(n) && i > n \implies \mathbb{1}_{i\le n} = 0 && (7.4.273)\\
&= \sum_{i=1}^{\infty}\sum_{n=0}^{\infty} \mathbb{1}_{i\le n} p(n) = \sum_{i=1}^{\infty}\sum_{n=i}^{\infty} p(n) && n < i \implies \mathbb{1}_{i\le n} = 0 && (7.4.274)\\
&= \sum_{i=1}^{\infty}\sum_{n=i}^{\infty} (1-\rho)\rho^n && p(n) = (1-\rho)\rho^n && (7.4.275)\\
&= \sum_{i=1}^{\infty}\sum_{n=0}^{\infty} (1-\rho)\rho^{n+i} && n \to n+i && (7.4.276)\\
&= \sum_{i=1}^{\infty} (1-\rho)\rho^i \sum_{n=0}^{\infty} \rho^n && \rho^{n+i} = \rho^i\rho^n && (7.4.277)\\
&= \sum_{i=1}^{\infty} (1-\rho)\rho^i \frac{1}{1-\rho} = \sum_{i=1}^{\infty} \rho^i = \sum_{i=0}^{\infty} \rho^{i+1} && i \to i+1 && (7.4.278)\\
&= \rho \sum_{i=0}^{\infty} \rho^i = \frac{\rho}{1-\rho}. && && (7.4.279)
\end{aligned}$$

Note that, since the summands are positive, we can use Fubini's theorem to justify the interchange of the summations.

s.6.1.12. Differentiate the left- and RHS of $(1-\rho)^{-1} = \sum_{n=0}^{\infty} \rho^n$ with respect to ρ and then multiply with ρ to get

$$\frac{\rho}{(1-\rho)^2} = \sum_{n=0}^{\infty} n\rho^n. \tag{7.4.280}$$

Then multiply both sides by $1-\rho$ and use that $p(n) = (1-\rho)\rho^n$ to get $\mathsf{E}[L]$.

Differentiating and multiplying with ρ a second time yields

$$\begin{aligned}
\rho\frac{(1-\rho)^2 + \rho 2(1-\rho)}{(1-\rho)^4} &= \rho\frac{1-2\rho+\rho^2+2\rho-2\rho^2}{(1-\rho)^4} = \rho\frac{1-\rho^2}{(1-\rho)^4} && (7.4.281)\\
&= \rho\frac{1+\rho}{(1-\rho)^3} = \sum_{n=0}^{\infty} n^2\rho^n, && (7.4.282)
\end{aligned}$$

and hence for $\mathsf{E}[L^2]$,

$$\begin{aligned}
(1-\rho)\sum_{n=0}^{\infty} n^2\rho^n &= \rho\frac{1+\rho}{(1-\rho)^2} = \frac{\rho}{(1-\rho)^2} + \frac{\rho^2}{(1-\rho)^2} = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} && (7.4.283)\\
&= \frac{2\rho^2}{(1-\rho)^2} + \rho\frac{(1-\rho)}{(1-\rho)^2} = \frac{2\rho^2}{(1-\rho)^2} + \frac{\rho}{1-\rho}. && (7.4.284)
\end{aligned}$$

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

s.6.2.1. With Little's law:

```
>>> labda = 5. # per minute
>>> W = 1.
>>> Q = labda*W
>>> Q
5.0
```

We can use the expression for $\mathsf{E}[W]$ to solve for ρ, but we also do a simple search.

```
>>> from scipy.optimize import bisect


>>> def find_W(rho): # return W -1 for given rho
...     ES = rho / labda
...     return rho / (1 - rho) * ES - 1
...

>>> rho = bisect(find_W, 0, 0.999)
>>> rho
0.8541019662513663
>>> ES = rho/labda
>>> ES
0.17082039325027326
>>> J = W + ES
>>> J
1.1708203932502732
>>> L = labda*J
>>> L
5.854101966251366
```

Next, find n such that $\sum_{j=0}^{n} p_j > 0.9$.

```
>>> n, total, p = 0, 0.0, 1 - rho
>>> while total <= 0.9:
...     total += p
...     n += 1
...     p *= rho
...
>>> total
0.9061042738302431
>>> n
15
```

Note that we increased n one time too often. As a check, use that $(1-\rho)\sum_{j=0}^{n} \rho^j = 1-\rho^{n+1}$.

```
>>> n -= 1 # get the right n
>>> 1-rho**(n) # this must be too small.
0.8900649689616529
```

```
>>> 1-rho**(n+1) # this must be OK.
0.9061042738302427
```

s.6.2.2. Start with the data, then do the computations.

```
>>> c = 2
>>> p = [0.4, 0.3, 0.2, 0.05, 0.05]
>>> labda = 10.0 / 60 # per hour

>>> EL = sum(n * p[n] for n in range(len(p)))
>>> EL
1.05
>>> EQ = sum(max(n - c, 0) * p[n] for n in range(len(p)))
>>> EQ
0.15000000000000002
>>> W = EQ / labda  # in minutes
>>> W
0.9000000000000001
>>> J = EL / labda  # in minutes
>>> J
6.300000000000001
>>> var_L = sum((n - EL) ** 2 * p[n] for n in range(len(p)))
>>> var_L
1.2475
>>> var_Q = sum((max(n - c, 0) - EQ) ** 2 * p[n] for n in range(len(p)))
>>> var_Q
0.22750000000000004
>>> ES = J - W
>>> rho = labda * ES / c
>>> rho
0.45
```

The expected number of busy servers is the average number in the system minus the average number in queue. It should also be equal to $\sum_{n=0}^{\infty} \min\{n,c\}p(n)$. Let's check.

```
>>> EBusy = EL - EQ
>>> EBusy
0.9
>>> u = sum(min(n, c) * p[n] for n in range(len(p)))
>>> u
0.8999999999999999
```

s.6.2.3. We import `numpy` and convert the lists to arrays is to fix the output precision to 3, otherwise we get long floats in the output.

First the case with $b = 2$.

```
>>> import numpy as np
>>> np.set_printoptions(precision=3)
```

```
>>> labda, mu, c = 6.0,  5.0, 1
>>> rho = labda / mu
>>> K = c + 2


>>> p = np.array([rho ** n for n in range(K + 1)]) # range(n) is up to n
>>> G = sum(p)
>>> 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
>>> Q = sum(max(n - c,0) * p[n] for n in range(len(p)))
>>> Q
0.9120715350223545
>>> lost = labda * p[-1]  # the last element of p
>>> labda - lost # accepted, hence served
4.06855439642325
```

Now increase the buffer b to 5.

```
>>> K = c + 5
>>> p = np.array([rho ** n for n in range(K + 1)]) # range(n) is up to n
>>> G = sum(p)
>>> p /= G  # normalize
>>> lost = labda * p[-1]  # the last element of P
>>> labda - lost # accepted, hence served
4.612880368265357
```

We see that since the server is overloaded, the acceptance is not much affected by increasing the buffer space. We need an extra server.

s.6.2.4. The computations..

```
>>> import numpy as np
>>> # from math import factorial

>>> labda = np.array([10.0, 15.0, 15.0, 10.0, 5.0])
>>> mu = np.array([0., 5., 10., 10., 10., 10.])
>>> c = 2
>>> K = 5

>>> p = np.ones_like(mu)
>>> for i in range(K):
...     p[i+1] = labda[i] *p[i]/ mu[i+1]
...
```

```
>>> p /= p.sum()
>>> p
array([0.058, 0.116, 0.174, 0.261, 0.261, 0.13 ])
>>> labdaBar = sum(labda[n] * p[n] for n in range(len(labda)))
>>> labdaBar
8.840579710144928
>>> L = sum(n * p[n] for n in range(len(p)))
>>> L
2.942028985507246
>>> Q = sum(max(n - c,0) * p[n] for n in range(len(p)))
>>> Q
1.1739130434782608
>>> J = L / labdaBar
>>> J  # time in the system
0.3327868852459016
>>> W = Q / labdaBar
>>> W  # time in queue
0.13278688524590163
```

s.6.2.5. What is the load?

```
>>> labda = 28.0 / 60  # arrivals per minute
>>> ES = 4.0
>>> labda * ES
1.8666666666666667
```

Clearly, we need at least two servers.

```
>>> from math import factorial


>>> def Q(labda, ES, c):
...     rho = labda * ES / c
...     G = sum([(c * rho) ** n / factorial(n) for n in range(c)])
...     G += (c * rho) ** c / (1.0 - rho) / factorial(c)
...     return (c * rho) ** c / (factorial(c) * G) * rho / (1.0 - rho) ** 2
...
>>> Q(labda, ES, c=2)/labda  # in minutes, Little's law
27.034482758620694
>>> Q(labda, ES, c=3)/labda  # in minutes, Little's law
1.3542675591474136
>>> Q(labda, ES, c=4)/labda  # in minutes, Little's law
0.26778942672317635
```

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. The average cost of workers waiting in queue is proportional to $\mathsf{E}[Q]$. At the crib, the load is $\lambda \mathsf{E}[S]$, hence, the average number of idle server is $c - \lambda \mathsf{E}[S]$.

```
>>> c = 2
>>> Q(labda, ES, c) + (c - labda * ES)
12.749425287356324
>>> c = 3
>>> Q(labda, ES, c) + (c - labda * ES)
1.7653248609354597
>>> c = 4
>>> Q(labda, ES, c) + (c - labda * ES)
2.2583017324708154
```

s.6.2.6. From the figure in the hint, the situation with the taxis corresponds to an $M/M/1$ queue, only the states have a 'different name'. Let l be the number of jobs in an M/M/1 queue. To make a mapping from l to the number of parties i and number of taxis j, observe that $l = 3 - j + i$ and $\mathbb{1}_{j>0}\mathbb{1}_{i>0} = 0$, because riders don't wait when there are taxis available. To see this, consider the following table

j	i	l
3	0	0
2	0	1
1	0	2
0	0	3
0	1	4
0	2	5

With this mapping, the expected number of taxis T is $\mathsf{E}[T] = \sum_{l=0}^{3}(3-l)p(l)$.

```
>>> labda = 12.0  # per hour
>>> mu = 15.0  # per hour
>>> rho = labda / mu
>>> n = 3  # number of taxis

>>> ET, p = 0, 1 - rho
>>> for l in range(n):
...     ET += (n - l) * p
...     p *= rho
...
>>> ET
1.0479999999999998
```

For the expected number of groups waiting note that $l = 3 - j + i$ always. Taking expectations, we see that $\mathsf{E}[L] = 3 - \mathsf{E}[T] + \mathsf{E}[G]$, where $\mathsf{E}[G]$ is the expected number of groups.

```
>>> EL = rho/(1-rho)
>>> EG = EL-n + ET
>>> EG
2.048000000000001
```

Computing the waiting times is tricky. For the taxis, the rate at which 'jobs' arrive, is the arrival rate of the riders. For the riders, the rate at which 'jobs' arrive is the arrival rate of taxis.

```
>>> ET/labda # Waiting for taxis
0.08733333333333332
>>> EG/mu # waiting time for groups
0.13653333333333334
```

What would be the impact of allowing 4 cabs?

```
>>> n = 4  # number of taxis

>>> ET, p = 0, 1 - rho
>>> for l in range(n):
...     ET += (n - l) * p
...     p *= rho
...
>>> ET
1.6383999999999999
>>> EG = EL-n + ET
>>> EG
1.6384000000000007
```

s.6.2.7. This is a case with $n=0$, hence $\mathsf{E}[G] = \mathsf{E}[L]$.

s.6.2.8. We concentrate on departure epochs of the taxis. Thus, the kth period is the time between the departure of taxi $k-1$ and taxi k. During the kth epoch a_k batches can arrive. The system starts with a_0 batches in queue.

More generally, consider the arrival of taxi k. Let this taxi see $Q_{s,k}$ riders at the head of the line. Let b be the index of the first group in the queue, hence group $b-1$ stands at the head of the line. Then,

$$
\begin{aligned}
d_k &= \min\{Q_{s,k}, 4\}, && \text{riders served by taxi } k, && (7.4.285)\\
Q'_s &= Q_{s,k} - d_k, && \text{riders remaining behind at head of line}, && (7.4.286)\\
Q'_k &= Q_k + a_{k+1}, && \text{groups just before arrival taxi } k+1, && (7.4.287)\\
h_k &= \mathbb{1}_{Q'_s>0}\mathbb{1}_{Q'_k>0} && \text{If } h_k = 1 \text{ a group can move to the head of the line}, && (7.4.288)\\
Q_{k+1} &= Q'_k - h_k && \text{queue of groups seen by taxi } k+1, && (7.4.289)\\
Q_{s,k+1} &= Q'_s(1-h_k) + h_k B_b && \text{riders at head of the line seen by taxi } k+1, && (7.4.290)\\
b &= b + h_k && \text{increase index of served batches by one}, && (7.4.291)\\
& && && (7.4.292)
\end{aligned}
$$

s.6.3.1. Start with the simple case. $B \equiv 2 \implies \mathsf{V}[B] = 0 \implies C_s^2 = 0$, $\rho = \lambda \mathsf{E}[B]\mathsf{E}[S] = 1 \cdot 2 \cdot 25/60 = 5/6$. Hence,

$$
\mathsf{E}[L] = \frac{1}{2}\frac{5/6}{1/6}2 + \frac{1}{2}\frac{5/6}{1/6} = 5 + \frac{5}{2}. \tag{7.4.293}
$$

Now the other case. $\mathsf{E}[B^2] = (1+4+9)/3 = 14/3$. Hence, $\mathsf{V}[B] = 14/3 - 4 = 2/3$. Hence, $C_s^2 = \frac{1}{6}$. And thus,

$$
\mathsf{E}[L] = \frac{1+1/6}{2}\frac{5/6}{1/6}2 + \frac{1}{2}\frac{5/6}{1/6} = \frac{7}{6}5 + \frac{5}{2}. \tag{7.4.294}
$$

The ratio between $\mathsf{E}[L]$ is 10/9. A reduction of about 10% in waiting time can achieved by working in fixed batch sizes.

s.6.3.2.

$$\frac{\mathsf{E}[W(M^X/M/1)]}{\mathsf{E}[W(M/M/1)]} = \frac{\mathsf{E}[L_s(M^X/M/1)]}{\mathsf{E}[L_s(M/M/1)]} = \frac{\mathsf{E}[L_s(M^X/M/1)]}{\rho} = \frac{\mathsf{E}[B^2]}{2\,\mathsf{E}[B]} + \frac{1}{2}. \tag{7.4.295}$$

The RHS is ≥ 1, because $\mathsf{V}[B] \geq 0 \implies \mathsf{E}[B]^2 \geq (\mathsf{E}[B])^2$. Clearly, if variability increases, the average waiting time increases

s.6.3.3. We need $\mathsf{V}[B]$ and $\mathsf{E}[B]$. For this,

$$M_B(s) = \mathsf{E}\left[e^{sB}\right] = \sum_{k=0}^{\infty} e^{sk}\,\mathsf{P}\{B=k\} \tag{7.4.296}$$

$$= \sum_{k=0}^{\infty} e^{sk} p q^{k-1} = \frac{p}{q}\sum_{k=0}^{\infty}(qe^s)^k = \frac{p}{q}\frac{1}{1-qe^s}, \tag{7.4.297}$$

$$\mathsf{E}[B] = M_B'(0) = \frac{p}{q}\frac{q}{(1-qe^s)^2}\bigg|_{s=0} = \frac{p}{(1-q)^2} = \frac{1}{p}, \tag{7.4.298}$$

$$\mathsf{E}[B^2] = M_B''(0) = \frac{2}{p^2} - \frac{1}{p}, \tag{7.4.299}$$

$$\mathsf{V}[B] = \mathsf{E}[B^2] - (\mathsf{E}[B])^2 = \frac{2}{p^2} - \frac{1}{p} - \frac{1}{p^2} = \frac{1}{p^2} - \frac{1}{p}, \tag{7.4.300}$$

$$C_s^2 = \frac{\mathsf{V}[B]}{(\mathsf{E}[B])^2} = p^2\left(\frac{1}{p^2} - \frac{1}{p}\right) = 1-p, \tag{7.4.301}$$

$$(1+C_s^2)/2 = 1 - p/2, \tag{7.4.302}$$

$$\mathsf{E}[L] = \left(1-\frac{p}{2}\right)\frac{\rho}{1-\rho}\frac{1}{p} + \frac{1}{2}\frac{\rho}{1-\rho} = \frac{\rho}{1-\rho}\frac{1}{p}. \tag{7.4.303}$$

$\mathsf{E}[B] = 1 \implies p = 1 \implies \mathsf{E}[L] = \rho/(1-\rho)$.

s.6.3.4. Suppose the kth job has batch size B, then

$$X_k = \int_{D_{k-1}}^{D_k} L_s^B(s)\,ds = BS_1 + (B-1)S_2 + \cdots + S_B. \tag{7.4.304}$$

With this, and independence of B and S,

$$\mathsf{E}[X] = \mathsf{E}[BS_1 + (B-1)S_2 + \cdots + S_B] = \mathsf{E}\left[\sum_{j=1}^{B} jS_{B+1-j}\right] \tag{7.4.305}$$

$$= \mathsf{E}\left[\sum_{j=1}^{B} j\right]\mathsf{E}[S] = \mathsf{E}[B(B+1)/2]\,\mathsf{E}[S]. \tag{7.4.306}$$

s.6.3.5. From [6.3.4], rate stability ($\delta = \lambda$), $\rho = \lambda\,\mathsf{E}[B]\,\mathsf{E}[S]$,

$$\mathsf{E}\left[L_s^B\right] = Y = \delta X = \lambda\frac{\mathsf{E}[B^2] + \mathsf{E}[B]}{2}\mathsf{E}[S] = \lambda\frac{\mathsf{E}[B^2]}{2}\mathsf{E}[S] + \frac{\rho}{2}. \tag{7.4.307}$$

s.6.3.6. We have

$$\frac{\mathsf{E}[B^2]}{(\mathsf{E}[B])^2} = \frac{\mathsf{E}[B^2] - (\mathsf{E}[B])^2 + (\mathsf{E}[B])^2}{(\mathsf{E}[B])^2} = \frac{\mathsf{V}[B] + (\mathsf{E}[B])^2}{(\mathsf{E}[B])^2} = C_s^2 + 1. \tag{7.4.308}$$

s.6.3.7.

$$\begin{aligned}\sum_{i=1}^{\infty} iG(i-1) &= \sum_{i=0}^{\infty}(i+1)G(i) = \sum_{i=0}^{\infty} iG(i) + \sum_{i=0}^{\infty} G(i) \\ &= (\mathsf{E}[B^2] - \mathsf{E}[B])/2 + \mathsf{E}[B].\end{aligned} \tag{7.4.309}$$

s.6.4.1. $\rho = \lambda\,\mathsf{E}[S] = (3/8)\cdot 2 = 3/4$, $\mathsf{E}[W] = 4.5$ h. If we were able to reduce all service variability, i.e., $C_s^2 = 0$, then still $\mathsf{E}[W] = 3$h. Hence, we have to increase capacity, or reduce $\mathsf{E}[S]$. Another possibility is to plan the arrival of jobs such that $C_a^2 = 0$. However, typically this is not possible. Would you accept that the supermarket plans your visits?

s.6.4.2. $\mathsf{V}[S] = 0 \implies C_s^2 = 0 \implies \mathsf{E}[W(M/D/1)] = \mathsf{E}[W(M/M/1)]/2$.

s.6.4.3.

$$\mathsf{E}[S] = \alpha/2, \qquad \mathsf{E}[S^2] = \int_0^{\alpha} x^2\,\mathrm{d}x/\alpha = \alpha^2/3, \tag{7.4.310}$$

$$\mathsf{V}[S] = \alpha^2/3 - \alpha^2/4 = \alpha^2/12, \qquad C_s^2 = (\alpha^2/12)/(\alpha^2/4) = 1/3, \tag{7.4.311}$$

$$\rho = \lambda\alpha/2, \tag{7.4.312}$$

$$\mathsf{E}[W] = \frac{1+C_s^2}{2}\frac{\lambda\alpha/2}{1-\lambda\alpha/2}\frac{\alpha}{2}, \qquad \mathsf{E}[J] = \mathsf{E}[W] + \frac{\alpha}{2}. \tag{7.4.313}$$

s.6.4.4.

$$\lambda\pi(0)\,\mathsf{E}[S] = 1-\pi(0) \iff \pi(0) = \frac{1}{1+\lambda\,\mathsf{E}[S]} \iff 1-\pi(0) = \frac{\lambda\,\mathsf{E}[S]}{1+\lambda\,\mathsf{E}[S]}. \tag{7.4.314}$$

There is another way to derive this. The system can contain at most 1 job. Necessarily, if the system contains a job, this job must be in service. All jobs that arrive while the server is busy are rejected. Just after a departure, the average time until the next arrival is $1/\lambda$, and then a new service starts with an average duration of $\mathsf{E}[S]$. After this departure, a new cycle starts. Thus, $\rho = \mathsf{E}[S]/(1/\lambda + \mathsf{E}[S]) = \lambda\,\mathsf{E}[S]/(1+\lambda\,\mathsf{E}[S])$.

For the $G/G/1/1$ queue the PASTA property does not hold, hence $p(0) \neq \pi(0)$ in general.

s.6.4.5. Right after the server becomes free, the time to a new arrival is $\sim \mathrm{Exp}(\lambda)$, with mean $1/\lambda$. For $\mathsf{E}[U]$, solve the expression of the hint with $\mathsf{E}[I] = 1/\lambda$. To see why $\rho = \mathsf{E}[U]/(\mathsf{E}[I] + \mathsf{E}[U])$, apply the renewal reward equation. Take $Y(t) = \int_0^t \mathbb{1}_{L_s(s)=1}\,\mathrm{d}t$. Taking as sampling epochs the departures that leave an empty system behind, $X_k = U_k$ and $T_k = I_k + U_k$, where U_k is the kth busy time, and I_k the kth idle time. Then $Y(t)/ \to \rho$, $X_k/k \to \mathsf{E}[U]$, and $k/(I_k + U_k) \to 1/(\mathsf{E}[I] + \mathsf{E}[U])$, which is the λ we use in the renewal-reward theorem.

We can also obtain $\mathsf{E}[U]$ by means of a recursion. The first customer starts a busy time of average duration $\mathsf{E}[S]$. However, during this service $\lambda\,\mathsf{E}[S]$ new jobs arrive, in expectation. Each of these jobs restarts the busy-period. Hence, $\mathsf{E}[U] = \mathsf{E}[S] + \lambda\,\mathsf{E}[S]\,\mathsf{E}[U]$.

s.6.4.6. At time s, the number of departures is $D(s)$. Thus, $D(s)+1$ is the first job to depart after time s. The departure time of this job is $D_{D(s)+1}$, hence the remaining service time at time s is $D_{D(s)+1} - s$, provided this job is in service.

s.6.4.7.

$$\lambda\,\mathsf{E}[S^2] = \frac{\mathsf{E}[S^2]}{(\mathsf{E}[S])^2}\lambda(\mathsf{E}[S])^2 = \frac{\mathsf{E}[S^2]}{(\mathsf{E}[S])^2}\rho\,\mathsf{E}[S] = (1+C_s^2)\rho\,\mathsf{E}[S]. \tag{7.4.315}$$

s.6.4.8. The probability to find the server busy upon arrival is ρ, and $S_r > 0 \iff L > 0$. Therefore,

$$\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \mid S_r > 0] + (1-\rho)\,\mathsf{E}[S_r \mid S_r = 0] = \rho\,\mathsf{E}[S_r \mid S_r > 0]. \tag{7.4.316}$$

s.6.4.9. For the $M/M/1$, service times are memoryless, hence, $\mathsf{E}[S_r \mid S_r > 0] = \mathsf{E}[S]$. But, from [6.4.8], $\mathsf{E}[S_r] = \rho\,\mathsf{E}[S_r \mid S_r > 0]$ for the $M/G/1$ queue. The difference is due to the fact that only jobs that arrive at a busy system see $S_r > 0$.

s.6.4.10. $\rho = \sum_{i=1}^{\infty} p(i) = \sum_{i=1}^{\infty} \pi(i) = \sum_{i=1}^{\infty} \delta(i)$.

s.6.5.1. In the $M/M/1$ queue, $G(0) = 1$ and $G(1) = G(2) = \cdots = 0$. Thus, $\sum_{m=0}^{n} G(n-m)\pi(m) = G(0)\pi(n) = \pi(n)$.

s.6.5.2. 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 6.3 to see that

$$\begin{aligned}
\mu\,\mathsf{E}[L] &= \sum_{n=0}^{\infty} n\,\mu\pi(n), \quad \text{now substitute for } \mu\pi(n) \text{ the recursion (6.5.1)}, && (7.4.317)\\
&= \lambda \sum_{n=0}^{\infty} n \sum_{i=0}^{n-1} \pi(i) G(n-1-i) = \lambda \sum_{n=0}^{\infty} n \sum_{i=0}^{\infty} \mathbb{1}_{i<n} \pi(i) G(n-1-i) && (7.4.318)\\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} \mathbb{1}_{i<n} n G(n-1-i) = \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=i+1}^{\infty} n G(n-1-i) && (7.4.319)\\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} (n+i+1) G(n) = \lambda \sum_{i=0}^{\infty} \pi(i) \left[\sum_{n=0}^{\infty} n G(n) + (i+1) \sum_{n=0}^{\infty} G(n) \right] && (7.4.320)\\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} n G(n) + \lambda\,\mathsf{E}[B] \sum_{i=0}^{\infty} \pi(i)(i+1) && (7.4.321)\\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \frac{\mathsf{E}\left[B^2\right] - \mathsf{E}[B]}{2} + \lambda\,\mathsf{E}[B](\mathsf{E}[L]+1) && (7.4.322)\\
&= \lambda \frac{\mathsf{E}\left[B^2\right] - \mathsf{E}[B]}{2} + \lambda\,\mathsf{E}[B]\,\mathsf{E}[L] + \lambda\,\mathsf{E}[B] && (7.4.323)\\
&= \lambda \frac{\mathsf{E}\left[B^2\right]}{2} + \lambda\,\mathsf{E}[B]\,\mathsf{E}[L] + \lambda \frac{\mathsf{E}[B]}{2}. && (7.4.324)
\end{aligned}$$

Dividing both sides by μ and using that $\lambda\,\mathsf{E}[B]/\mu = \rho$, we obtain

$$EL = \lambda \frac{\mathsf{E}\left[B^2\right]}{2} \mathsf{E}[S] + \rho\,\mathsf{E}[L] + \frac{\rho}{2}. \tag{7.4.325}$$

By bringing $\rho\,\mathsf{E}[L]$ to the LHS, the RHS becomes equal to (6.3.5).

s.6.5.3. Jobs can arrive in batches larger than 1, but items leave one by one.

s.6.5.4. For the partial acceptance case, any job is accepted, but the system only admits whatever fits. As level $n \in 0,1,\ldots,K-1$ is still up-crossed by any batch of size at least $n-m$ when the system is in state m, the formula for the up-crossing rate is identical to the case without this acceptance policy. Hence, $\mu\pi(n+1) = \lambda \sum_{m=0}^{n} \pi(m) G(n-m)$, for $n = 0,1,\ldots,K-1$.

s.6.5.5. The complete-acceptance policy is actually quite simple. As any batch will be accepted when $n \leq K$, the queue length is not bounded. Only when the number of items in the system is larger than K, we do not accept jobs.

$$\mu\pi(n+1) = \begin{cases} \lambda \sum_{m=0}^{n} \pi(m)G(n-m), & \text{for } n \leq K, \\ \lambda \sum_{m=0}^{K} \pi(m)G(n-m), & \text{for } n > K. \end{cases} \tag{7.4.326}$$

s.6.5.6. Suppose a batch of size k arrives when the system contains m items. When $m+k \leq K$, the batch can be accepted since the entire batch will fit into the queue, otherwise it will be rejected. Further, level $n, 0 \leq n < K$, can only be crossed when $m+k > n$. Thus, for $n = 0, \ldots, K-1$,

$$\mu\pi(n+1) = \lambda \sum_{m=0}^{n} \pi(m) \mathsf{P}\{n-m < B \leq k-m\} \tag{7.4.327}$$

$$= \lambda \sum_{m=0}^{n} \pi(m)[G(n-m) - G(K-m)], \tag{7.4.328}$$

Let us check this for some simple cases. First, when $K \to \infty$, then $G(K-m) \to 0$, so we get our earlier result. Second, take $n = K$. Then $G(n-m) - G(K-m) = 0$ for all m, so that the RHS is 0, as it should. Third, take $n = 0$ and $K = 1$, then $\mu\pi(1) = \lambda\pi(0)$. This also makes sense.

s.6.6.1. After job $k-1$ left, job k first has to arrive. Hence, $\mathsf{E}[D_k - D_{k-1}] = \mathsf{E}[X_k + S_k] = 1/\lambda + \mathsf{E}[S]$, where we use that X_k is memoryless.

s.6.6.2. Since $X \sim \text{Exp}(\lambda)$ and $S \sim \text{Exp}(\mu)$, and X and S are independent, their joint density is $f_{X,S}(x,y) = \lambda\mu e^{-\lambda x - \mu y}$. With this,

$$\mathsf{P}\{X+S \leq t\} = \lambda\mu \int_0^\infty \int_0^\infty e^{-\lambda x - \mu y} \mathbb{1}_{x+y \leq t} \, \mathrm{d}x \, \mathrm{d}y \tag{7.4.329}$$

$$= \lambda\mu \int_0^t \int_0^{t-x} e^{-\lambda x - \mu y} \, \mathrm{d}y \, \mathrm{d}x \tag{7.4.330}$$

$$= \lambda\mu \int_0^t e^{-\lambda x} \int_0^{t-x} e^{-\mu y} \, \mathrm{d}y \, \mathrm{d}x \tag{7.4.331}$$

$$= \lambda \int_0^t e^{-\lambda x}(1 - e^{-\mu(t-x)}) \, \mathrm{d}x \tag{7.4.332}$$

$$= \lambda \int_0^t e^{-\lambda x} \, \mathrm{d}x - \lambda e^{-\mu t} \int_0^t e^{(\mu-\lambda)x} \, \mathrm{d}x \tag{7.4.333}$$

$$= 1 - e^{-\lambda t} - \frac{\lambda}{\mu-\lambda} e^{-\mu t}(e^{(\mu-\lambda)t} - 1) \tag{7.4.334}$$

$$= 1 - e^{-\lambda t} - \frac{\lambda}{\mu-\lambda} e^{-\lambda t} + \frac{\lambda}{\mu-\lambda} e^{-\mu t} \tag{7.4.335}$$

$$= 1 - \frac{\mu}{\mu-\lambda} e^{-\lambda t} + \frac{\lambda}{\mu-\lambda} e^{-\mu t}. \tag{7.4.336}$$

$$\tag{7.4.337}$$

The density $f_{X+S}(t)$ is the derivative of this expression with respect to t, hence,

$$f_{X+S}(t) = \frac{\lambda\mu}{\mu-\lambda}e^{-\lambda t} - \frac{\mu\lambda}{\mu-\lambda}e^{-\mu t} \tag{7.4.338}$$
$$= \frac{\lambda\mu}{\lambda-\mu}(e^{-\mu t} - e^{-\lambda t}). \tag{7.4.339}$$
$$\tag{7.4.340}$$

Conditioning is much faster:

$$f_{X+S}(t) = \mathsf{P}\{X+S \in \mathrm{d}t\} = \int_0^t \mathsf{P}\{S+x \in \mathrm{d}t\}\,\mathsf{P}\{X \in \mathrm{d}x\} \tag{7.4.341}$$
$$= \int_0^t f_S(t-x)f_X(x)\,\mathrm{d}x = \int_0^t \mu e^{-\mu(t-x)}\lambda e^{-\lambda x}\,\mathrm{d}x \tag{7.4.342}$$
$$= \lambda\mu e^{-\mu t}\int_0^t e^{x(\mu-\lambda)}\,\mathrm{d}x = \frac{\lambda\mu}{\mu-\lambda}e^{-\mu t}\left(e^{(\mu-\lambda)t}-1\right). \tag{7.4.343}$$

s.6.6.3. Use conditional probability to see that

$$\mathsf{P}\{Y_n = j\} = \int_0^\infty e^{-\lambda x}\frac{(\lambda x)^j}{j!}F(\mathrm{d}x) = \int_0^\infty e^{-\lambda x}\frac{(\lambda x)^j}{j!}\mu e^{-\mu x}\,\mathrm{d}x = \frac{\mu}{j!}\lambda^j\int_0^\infty e^{-(\lambda+\mu)x}x^j\,\mathrm{d}x \tag{7.4.344}$$
$$= \frac{\mu}{j!}\left(\frac{\lambda}{\lambda+\mu}\right)^j\int_0^\infty e^{-(\lambda+\mu)x}((\lambda+\mu)x)^j\,\mathrm{d}x = \frac{\mu}{j!}\left(\frac{\lambda}{\lambda+\mu}\right)^j\frac{j!}{\lambda+\mu}. \tag{7.4.345}$$

Method 2. Consider the Poisson process with rate $\lambda+\mu$, and thin with probability $\mu/(\lambda+\mu)$. Then the probability that j 'failures' occur before a 'success' is precisely $\mathsf{P}\{Y = j\}$.

s.6.6.4. Take $\alpha = \lambda/(\lambda+\mu)$ so that $f(j) = (1-\alpha)\alpha^j$.

$$G(j) = \sum_{k=j+1}^\infty f(k) = (1-\alpha)\sum_{k=j+1}^\infty \alpha^k = (1-\alpha)\alpha^{j+1}\sum_{k=0}^\infty \alpha^k = \alpha^{j+1}. \tag{7.4.346}$$

s.6.6.5. Observe that $f(j) = (1-\alpha)\alpha^j$, and $G(j) = \alpha^{j+1}$. As the normalization factor cancels out, we write $\pi(n) = \rho^n$.

For $n = 0$: $f(0)\pi(1) = \pi(0)G(0) \iff (1-\alpha)\rho = 1\,\alpha$, and this checks with the hint. For $n \geq 1$:

$$(1-\alpha)\rho^{n+1} = \pi(0)G(n) + \sum_{m=1}^n \pi(m)G(n+1-m) = \alpha^{n+1} + \sum_{m=1}^n \rho^m\alpha^{n-m+2} \tag{7.4.347}$$
$$= \alpha^{n+1} + \alpha^{n+2}\sum_{m=1}^n(\rho/\alpha)^m = \alpha^{n+1} + \alpha^{n+1}\rho\sum_{m=0}^{n-1}(\rho/\alpha)^m = \alpha^{n+1} + \alpha^{n+1}\rho\frac{1-(\rho/\alpha)^n}{1-\rho/\alpha} \tag{7.4.348}$$
$$= \alpha^{n+1} - \alpha^{n+1}(1-(\rho/\alpha)^n), \quad \text{as } 1-\rho/\alpha = -\rho, \tag{7.4.349}$$
$$= \alpha^{n+1}(\rho/\alpha)^n = \alpha\rho^n. \tag{7.4.350}$$

Since $\rho = \alpha/(1-\alpha)$ we see that the left- and RHSs are the same.

s.7.1.1.

$$aq^2 + bq = \alpha a(q^2+2q+1) + \alpha b(q+1) + \beta a(q^2-2q+1) + \beta b(q-1) + hq/(\lambda+\mu). \tag{7.4.351}$$

Matching the coefficients of q^2, q

$$a = \alpha a + \beta a \qquad \Longrightarrow a = a, \tag{7.4.352}$$

$$b = \alpha a 2 + \alpha b - \beta a 2 + \beta b + h/(\lambda+\mu), \qquad \Longrightarrow a = \frac{h}{2}\frac{1}{\mu-\lambda}, \tag{7.4.353}$$

$$0 = \alpha a + \alpha b + \beta a - \beta b, \qquad \Longrightarrow b = \frac{a}{\beta-\alpha}. \tag{7.4.354}$$

s.7.1.2. On the one hand, the cost of the jobs in the system during one cycle must be $V(1)$. The duration of one cycle is $C(1)$. By the renewal-reward theorem, the time-average cost is then $V(1)/C(1)$. On the other hand, if the time-average number of jobs in the system is $\mathsf{E}[L]$, and each job pays h per unit time, the time-average cost must be $h\,\mathsf{E}[L]$.

s.7.1.3.

$$\frac{V(1)}{1/\lambda + T(1)} = \frac{a+b}{1/\lambda + 1/(\mu-\lambda)} = \left(\frac{h}{2}\frac{1}{\mu-\lambda} + \frac{h}{2}\frac{\lambda+\mu}{(\mu-\lambda)^2}\right)\frac{\lambda(\mu-\lambda)}{\mu} \tag{7.4.355}$$

$$= \frac{h}{2}\rho\left(1+\frac{\lambda+\mu}{\mu-\lambda}\right) = \frac{h}{2}\rho\frac{2\mu}{\mu-\lambda} = h\frac{\rho}{1-\rho}. \tag{7.4.356}$$

s.7.1.4. The cost up to the qth job is the cost up to the arrival of job $q-1$ plus the cost while there are $q-1$ jobs in the system. The time between the arrival of job $q-1$ and q is $1/\lambda$. $W(q) = h\sum_{i=1}^{q}(i-1)$.

s.7.2.1. When we switch on the server, the queue 'drains' at rate $\mu - \lambda > 0$, with $\mu = 1/\mathsf{E}[S]$. Consequently, no matter how large N, $T(N) < \infty$. And, whenever the system is empty, the stochastic process restarts. As such cycles start over and over again, and the queue length can never 'escape to infinity'.

s.7.2.2. The total number $A(t)$ of job that arrive during $[0,t]$ does not depend on N. Thus, in (5.1.3), $\sum_{k=1}^{A(t)} S_k$ does not depend on N. Now use rate-stability.

s.7.2.3. In the hint, the first equation is superfluous. In the second, bq cancels at both sides, by which we find a. The third now follows.

s.7.2.4. For b, using the expressions for $\mathsf{E}[Y]$ and $\mathsf{E}\left[Y^2\right]$,

$$
\begin{aligned}
b(1-\mathsf{E}[Y]) &= a(\mathsf{E}\left[Y^2\right]-2\mathsf{E}[Y]+1)+\frac{1}{2}h\lambda\mathsf{E}\left[S^2\right] && (7.4.357)\\
&= \frac{h\mathsf{E}[S]}{2(1-\mathsf{E}[Y])}(\mathsf{E}\left[Y^2\right]-2\mathsf{E}[Y]+1)+\frac{1}{2}h\lambda\mathsf{E}\left[S^2\right] && (7.4.358)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(\lambda^2\mathsf{E}\left[S^2\right]+\lambda\mathsf{E}[S]-2\lambda\mathsf{E}[S]+1+\lambda\mathsf{E}\left[S^2\right]\frac{1-\lambda\mathsf{E}[S]}{\mathsf{E}[S]}\right) && (7.4.359)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(\lambda^2\mathsf{E}\left[S^2\right]-\lambda\mathsf{E}[S]+1+\frac{\lambda\mathsf{E}\left[S^2\right]}{\mathsf{E}[S]}-\lambda^2\mathsf{E}\left[S^2\right]\right) && (7.4.360)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(1+\frac{\lambda\mathsf{E}\left[S^2\right]}{\mathsf{E}[S]}-\lambda\mathsf{E}[S]\right) && (7.4.361)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(1+\frac{\lambda\mathsf{E}\left[S^2\right]}{\mathsf{E}[S]}-\lambda\frac{(\mathsf{E}[S])^2}{\mathsf{E}[S]}\right) && (7.4.362)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(1+\lambda\frac{\mathsf{V}[S]}{\mathsf{E}[S]}\right) && (7.4.363)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(1+\lambda\frac{\mathsf{V}[S]}{(\mathsf{E}[S])^2}\mathsf{E}[S]\right) && (7.4.364)\\
&= \frac{h\mathsf{E}[S]}{2(1-\lambda\mathsf{E}[S])}\left(1+\rho C_s^2\right). && (7.4.365)
\end{aligned}
$$

Divide now both sides by $1-\mathsf{E}[Y]$.

s.7.2.5. For a, multiply the numerator and denominator by $\mu = 1/\mathsf{E}[S]$. For b, multiply by $\mu^2 = 1/(\mathsf{E}[S])^2$, use that $C_s^1 = 1$ because the service times are exponentially distributed, and note that

$$
\frac{1+\rho}{1-\rho} = \frac{\mu+\lambda}{\mu-\lambda}. \qquad (7.4.366)
$$

s.7.2.6. $h\mathsf{E}[L] = h\mathsf{E}[Q]+h\mathsf{E}[L_s]$. $\mathsf{E}[Q] = \lambda\mathsf{E}[W]$ and $\mathsf{E}[L_s] = \lambda\mathsf{E}[S]$.

s.7.2.7. Note first that $C(N) = N(1/\lambda+\mathsf{E}[S]/(1-\rho)) = N/(\lambda(1-\rho))$. Then,

$$
\begin{aligned}
\frac{V(N)+K+W(N)}{C(N)} &= \left(aN^2+bN+K+hN(N-1)/2\lambda\right)\frac{\lambda(1-\rho)}{N} && (7.4.367)\\
&= \frac{h}{2}\rho N+\frac{h}{2}\frac{\rho}{1-\rho}(1+\rho C_s^2)+\frac{h}{2}(N-1)(1-\rho)+K\frac{\lambda(1-\rho)}{N} && (7.4.368)\\
&= \frac{h}{2}\frac{\rho}{1-\rho}(1+\rho C_s^2)+\frac{h}{2}(N-1+\rho)+K\frac{\lambda(1-\rho)}{N} && (7.4.369)\\
&= \frac{h}{2}\frac{\rho}{1-\rho}(\rho+\rho C_s^2+1-\rho)+\frac{h}{2}(N-1+\rho)+K\frac{\lambda(1-\rho)}{N} && (7.4.370)\\
&= \frac{h}{2}\frac{\rho^2}{1-\rho}(1+C_s^2)+\frac{h}{2}\rho+\frac{h}{2}(N-1+\rho)+K\frac{\lambda(1-\rho)}{N} && (7.4.371)\\
&= \frac{h}{2}\frac{\rho^2}{1-\rho}(1+C_s^2)+h\rho+h\frac{N-1}{2}+K\frac{\lambda(1-\rho)}{N}. && (7.4.372)
\end{aligned}
$$

s.7.3.1.

$$f_D(t) = (1-\rho)f_{X+S}(t) + \rho\mu e^{-\mu t} = (1-\rho)\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \rho\mu e^{-\mu t} \tag{7.4.373}$$

$$= \left(1-\frac{\lambda}{\mu}\right)\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \rho\mu e^{-\mu t} \tag{7.4.374}$$

$$= \frac{\mu-\lambda}{\mu}\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \frac{\lambda}{\mu}\mu e^{-\mu t} = -\lambda\left(e^{-\mu t} - e^{-\lambda t}\right) + \lambda e^{-\mu t} \tag{7.4.375}$$

s.7.3.2. A job leaving station i has to go somewhere, to another station, perhaps to station i again, or leave the network. As $\sum_{i=1}^M P_{ij}$ is the probability to be sent to some station in the network, P_{i0} is the probability a job leaves the network from station i.

s.7.3.3.

$$P = \begin{pmatrix} \alpha & 1-\alpha \\ \beta_1 & \beta_2 \end{pmatrix}, \qquad (\lambda_1, \lambda_2) = (\gamma, 0) + (\lambda_1, \lambda_2)P. \tag{7.4.376}$$

Solving first for λ_2 leads to $\lambda_2 = (1-\alpha)\lambda_1 + \beta_2\lambda_2$, so that

$$\lambda_2 = \frac{1-\alpha}{1-\beta_2}\lambda_1. \tag{7.4.377}$$

Next, using this and that $\lambda_1 = \alpha\lambda_1 + \beta_1\lambda_2 + \gamma$ gives

$$\begin{aligned} \gamma &= \lambda_1(1-\alpha) - \beta_1\lambda_2 = \lambda_1\left(1-\alpha-\beta_1\frac{1-\alpha}{1-\beta_2}\right) \\ &= \lambda_1(1-\alpha)\left(1-\frac{\beta_1}{1-\beta_2}\right) = \lambda_1(1-\alpha)\frac{1-\beta_1-\beta_2}{1-\beta_2}. \end{aligned} \tag{7.4.378}$$

Hence,

$$\lambda_1 = \frac{\gamma}{1-\alpha}\frac{1-\beta_2}{1-\beta_1-\beta_2}, \qquad \lambda_2 = \frac{1-\alpha}{1-\beta_2}\lambda_1 = \frac{\gamma}{1-\beta_1-\beta_2}. \tag{7.4.379}$$

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.7.3.4. The rate into state $(0,0)$ is $\mu_2 p(0,1) = \mu_2\rho_2$. The rate out of state $(0,0)$ is $\lambda p(0,0) = \lambda$. Since $\rho_2 = \lambda/\mu_2$, these two rates are the same.

s.7.3.5.

$$\text{rate out } = (\lambda+\mu_1)p(i,0) = \mu_1\rho_1^i + \lambda\rho_1^i = \lambda\rho_1^{i-1} + \mu_2\rho_1^i\rho_2 \tag{7.4.380}$$

$$= \lambda p(i-1,0) + \mu_2 p(i,1) = \text{ rate in.} \tag{7.4.381}$$

s.7.3.6. We show that the rate out is the rate in.

$$\text{rate out } = (\lambda+\mu_2)p(0,j) = \mu_2\rho_2^j + \lambda\rho_2^j = \mu_1\rho_1\rho_2^{j-1} + \mu_2\rho_2^{j+1} \tag{7.4.382}$$

$$= \mu_1 p(1,j-1) + \mu_2 p(0,j+1) = \text{ rate in.} \tag{7.4.383}$$

s.7.4.1. $P_{i,i+1} = 1$, and $P_{ij} = 0$ for $j \neq i+1$. As P is an upper-triangular matrix of dimension $M \times M$, $P^M = 0$.

s.7.4.2. P transient $\implies Q_{i0} = P^M_{i0} > 0 \implies \sum_{j=1}^M Q_{ij} < 1$. Since M is a finite number, there exists an $\epsilon > 0$ such that $\max\{\sum_{j=1}^M Q_{ij}\} < 1-\epsilon$. Writing $\mathbf{1}$ for the vector $(1,1,\ldots,1)$, this means that $Q\mathbf{1} < (1-\epsilon)\mathbf{1}$. But then, $Q^2\mathbf{1} = Q(Q\mathbf{1}) < (1-\epsilon)Q\mathbf{1} < (1-\epsilon)^2\mathbf{1}$. As $Q \geq 0$ element wise, we see that $0 \leq Q^n < (1-\epsilon)^n$. And then $P^n = Q^{n/M} < (1-\epsilon)^{n/M}$.

s.7.4.3. The above argumentation is not necessarily valid for matrices P that are infinite, since $\inf\{P^M_{ik}\}$ need not be strictly positive for all i,j.

s.7.4.4. The probability to move from any state n straightaway to 0 is $\beta^n > 0$. (Why do we require n to be finite?)

BIBLIOGRAPHY

S. Asmussen. *Applied Probability and Queues*. Springer-Verlag, Berlin, 2003.

J.C. Baez and J. Biamonte. Quantum techniques for stochastic mechanics, 2019.

M. Capiński and T. Zastawniak. *Probability through Problems*. Springer Verlag, 2nd edition, 2003.

D.R. Cox, editor. *Renewal Theory*. John Wiley & Sons Inc, New York, 1962.

P. Doyle and J Laurie Snell. *Random Walks and Electrical Networks*. Mathematical Association of America, 1984.

M. El-Taha and S. Stidham Jr. *Sample-Path Analysis of Queueing Systems*. Kluwer Academic Publishers, 1998.

J.R. Norris. *Markov Chains*. Cambridge University Press, 1997.

A.A. Yushkevich and E.B. Dynkin. *Markov Processes: Theorems and Problems*. Plenum Press, 1969.

NOTATION

$$a_k = \text{Number of arrivals in } k\text{th period} \tag{7.4.384}$$
$$A(t) = \text{Number of arrivals in } [0,t] \tag{7.4.385}$$
$$A_k = \text{Arrival time of } k\text{th job} \tag{7.4.386}$$
$$B = \text{General batch size} \tag{7.4.387}$$
$$B_k = \text{Batch size of } k\text{th job} \tag{7.4.388}$$
$$\tilde{A}_k = \text{Start of service of } k\text{th job} \tag{7.4.389}$$
$$c_k = \text{Service/production capacity in } k\text{th period} \tag{7.4.390}$$
$$d_k = \text{Number of departures in } k\text{th period} \tag{7.4.391}$$
$$c = \text{Number of servers} \tag{7.4.392}$$
$$C_a^2 = \text{Squared coefficient of variation of inter-arrival times} \tag{7.4.393}$$
$$C_s^2 = \text{Squared coefficient of variation of service times} \tag{7.4.394}$$
$$D(t) = \text{Number of departures in } [0,t] \tag{7.4.395}$$
$$D_k = \text{Departure time of } k\text{th job} \tag{7.4.396}$$
$$\delta = \text{Departure rate} \tag{7.4.397}$$
$$F = \text{Distribution of service time of a job} \tag{7.4.398}$$
$$I = \text{Idle time of single server} \tag{7.4.399}$$
$$J_k = \text{Sojourn time of } k\text{th job} \tag{7.4.400}$$
$$L(t) = \text{Number of customers/jobs in system at time } t \tag{7.4.401}$$
$$L_k = \text{Number in system as end of } k\text{th period} \tag{7.4.402}$$
$$L_s(t) = \text{Number of customers/jobs in service at time } t \tag{7.4.403}$$
$$\lambda = \text{Arrival rate} \tag{7.4.404}$$
$$\mu = \text{Service rate} \tag{7.4.405}$$
$$N(t) = \text{Number of arrivals in } [0,t] \tag{7.4.406}$$
$$N(s,t) = \text{Number of arrivals in } (s,t] \tag{7.4.407}$$
$$p(n) = \text{Long-run time average that system contains } n \text{ jobs} \tag{7.4.408}$$
$$\pi(n) = \text{Stationary probability that an arrival sees } n \text{ jobs in system} \tag{7.4.409}$$
$$Q(t) = \text{Number of customers/jobs in queue at time } t \tag{7.4.410}$$
$$Q_k = \text{Queue length as seen by } k\text{th job, or at end } k\text{th period} \tag{7.4.411}$$
$$\rho = \text{utilization of a single server} \tag{7.4.412}$$
$$S = \text{Generic service time} \tag{7.4.413}$$
$$S_k = \text{Service time of } k\text{th job} \tag{7.4.414}$$
$$U = \text{Busy time of single server} \tag{7.4.415}$$
$$W = \text{Generic waiting time (in queue)} \tag{7.4.416}$$
$$W_k = \text{Waiting time in queue of } k\text{th job} \tag{7.4.417}$$
$$X = \text{Generic inter-arrival time between two consecutive jobs} \tag{7.4.418}$$
$$X_k = \text{Inter-arrival time between job } k-1 \text{ and job } k \tag{7.4.419}$$
$$\tag{7.4.420}$$

INDEX