Analysis of Queueing Systems with Sample Paths and Simulation

Nicky D. van Foreest

February 17, 2017

Contents

1 Introduction

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

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

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

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

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

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

2. The fraction of customers $\pi(n)$ that 'see upon arrival' the system with n customers. This measure relates to customer perception and lost sales, i.e., fractions of arriving customers that do not enter the system.

3. The average, variance, and/or distribution of the waiting time.

4. The average, variance, and/or distribution of the number of customers in the system.

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

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

In these notes we will be primarily concerned with making models of queueing systems such that we can compute or estimate the above performance measures. Part of our work is to derive analytic models. The benefit of such models is that they offer structural insights into the behavior of the system and scaling laws, such as that the average waiting time scales (more or less) linearly in the variance of the service times of individual customers. However, these models have severe shortcomings when it comes to analyzing real queueing systems, in particular when particular control rules have to be assessed. Consider, for example, the service process at a check-in desk of KLM. Business customers and economy customers are served by two separate queueing systems. The business customers are served by one server, server A say, while the economy class customers by three servers, say. What would happen to the sojourn time of the business customers if server A would be allowed to serve economy class customers when the business queue is empty? For the analysis of such cases simulation is a very useful and natural approach.

In the first part of these notes we concentrate on the analysis of *sample paths of a queueing process*. We assume that a typical sample path captures the 'normal' stochastic behavior of the system. This sample-path approach has two advantages. In the first place, most of the theoretical results follow from very concrete aspects of these sample paths. Second, the analysis of sample-paths carries over right away to simulation. In fact, simulation of a queueing system offers us one (or more) sample paths, and based on such sample paths we derive behavioral and statistical properties of the system. Thus, the performance measures defined for sample paths are precisely those used for simulation. Our aim is not to provide rigorous proofs for all results derived below; for the proofs and further background discussion we refer to El-Taha and Stidham Jr. [1998]. As a consequence we tacitly assume in the remainder that results derived from the (long-run) analysis of a particular sample path are equal to their 'probabilistic counterpart'.

In the second part we construct algorithms to analyze open and closed queueing networks. Many of the sample path results developed for the single-station case can be applied to these networks. As such, theory, simulation and algorithms form a nicely round out part of work. For this part we refer to book of Prof. Zijm; the present set of notes augment the discussion there.

I urge you to make *all* exercises in this set of notes. Many exercises require many of the tools you learned previously in courses on calculus, probability, and linear algebra. Here you can see them applied. Moreover, many of these tools will be useful for other, future, courses. Thus, the investments made here will pay off for the rest of your (student) carreer. Moreover, the exercises are meant to *illustrate* the material and to force you to *think* about it. Thus, the main text does not contain many examples; the exercises form the examples.

You'll notice that many of these problems are quite difficult, often not because the problem itself is difficult, but because you need to combine a substantial amount of knowledge all at the

same time. All this takes time and effort. Next to this, I did not include the exercises with the intention that you would find them easy. The problems should be doalbe, but hard.

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

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

The meaning of the symbols in the margin of pages are as follows:

- The symbol in the margin means that you have to memorize the *emphasized concepts*.

- The symbol in the margin means that this question has a *hint*.

- The symbol that this question or its solution requires still some *work on my part*; you can skip it.

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

2 Single-Station Queueing Systems

In this chapter, we start with a discussion of the exponential distribution and the related Poisson process, as these concepts are perhaps the most important building blocks of queueing theory. With these concepts we can specify the arrival and service processes of customers, so that we can construct queueing processes and define performance measures to provide insight into the (transient and average) behavior of queueing processes. As it turns out, these constructions can be easily implemented as computer programs, thereby allowing to use simulation to analyze queueing systems. We then continue with developing models for various single-station queueing systems in steady-state, which is, in a sense to be discusses later, the long-run behavior of a stochastic system.[1] In the analysis we use sample-path arguments to count how often certain events occur as functions of time. Then we define probabilities in terms of limits of fractions of these counting processes. Another useful aspect of sample-path analysis is that the definitions for the performance measures are entirely constructive, hence by leaving out the limits, they provide expressions that can be right away used in statistical analysis of (simulations of) queueing systems. Level-crossing arguments will be of particular importance as we use these time and again to develop recursions by which we can compute steady-state probabilities of the queue length or waiting time process.

2.1 Exponential Distribution

As we will see in the sections to come, the modeling and analysis of any queueing system involves the specification of the (probability) distribution of the time between consecutive arrival epochs of jobs, or the specification of the distribution of the number of jobs that arrive in a certain interval. For the first case, the most common distribution is the exponential distribution, while for the second it is the Poisson distribution. For these reasons we start our discussion of the analysis of queueing system with these two exceedingly important distributions. In the ensuing sections we will use these distributions time and again.

As mentioned, one of the most useful models for the inter-arrival times of jobs assumes that the sequence $\{X_i\}$ of inter-arrival times is a set of *independent and identically distributed (i.i.d.)* random variables. Let us write X for the generic random time between two successive arrivals. For many queueing systems, measurements of the inter-arrival times between consecutive arrivals show that it is reasonable to model an inter-arrival X as an *exponentially distributed* random variable, i.e.,

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

The constant λ is often called the *rate*. The reason behind this will be clarified once we relate the exponential distribution to the Poisson process in Section 2.2. In the sequel we often write $X \sim \exp\lambda$ to mean that X is exponentially distributed with rate λ.

Let us show with simulation how the exponential distribution originates. Consider N people that regularly visit a shop. We assume that we can characterize the interarrival times $\{X_k^i, k =$

[1] This statement is, admittedly, vague, to say the least.

$1,2,\ldots\}$ of customer i by some distribution function, for instance the uniform distribution. Then,

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

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

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

must be the inter-arrival time between the $k-1$th and kth visit to the shop. Thus, starting from interarrival times of individual customers we constructed interarrival times as seen by the shop.

To plot the *empirical distribution function*, or the histogram, of the inter-arrival times at the shop, consider

$$\mathbb{P}_n(X \le x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{X_i \le x},$$

where[2] $\mathbb{1}_{X_i \le x} = 1$ if $X_i \le x$ and $\mathbb{1}_{X_i \le x} = 0$ if $X_i > x$. Thus, for a simulation of duration n, this formula counts all interarrival times that are smaller than x.

We now compare $\mathbb{P}_n$ to the density of the exponential distribution, i.e., to $\lambda e^{-\lambda t}$ several simulation scenarios. As a first example, take $N = 1$ and let the computer generate $n = 100$ uniformly distributed numbers on the set $[4,6]$. Thus, the time between two visits of an individual customer is somewhere between 4 and 6 hours. In a second simulation we take $N = 3$, and in the third, $N = 10$. The emperical distributions are shown, from left to right, in the three panels in Figure 2.1. The continuous curve is the graph of $\lambda e^{-\lambda x}$ where $\lambda = N/5$. Recall from Eq. (2.19) that when N persons visits the shop, each with an average interarrival time of 5 hours, it must be that the arrival rate is $N/5$. As a second example we take the inter-arrival times to be normally distributed times with mean 5 and $\sigma = 1$. The results are shown in the second row of Figure 2.1.

As the graphs show, even when the customer population consists of 10 members each visiting the shop with an inter-arrival time that is quite 'far' from exponential, the distribution of the inter-arrival times as observed by the shop is very well approximated by an exponential distribution. Thus, for a real shop, with many thousands of customers, or a hospital, call center, in fact for nearly every system that deals with random demand, it seems reasonable to use the exponential distribution to model inter-arrival times. In conclusion, the main conditions to use an exponential distribution are: 1) arrivals have to be drawn from a large population, and 2) each of the arriving customers decides, independent of the others, to visit the system.

Another reason to use the exponential distribution is that an exponentially distributed random variable is *memoryless*, that is, X is memoryless if it satisfies the property that

$$\mathbb{P}\{X > t + h | X > t\} = \mathbb{P}\{X > h\}.$$

In words, the probability that X is larger than some time $t+h$, conditional on it being larger than a time t, is equal to the probability that X is larger than h. Thus, no matter how long we

[2] More generally, $\mathbb{1}_A$ is the indicator function of the event A meaning that $\mathbb{1}_A = 1$ if A is true and $\mathbb{1}_A = 0$ otherwise.

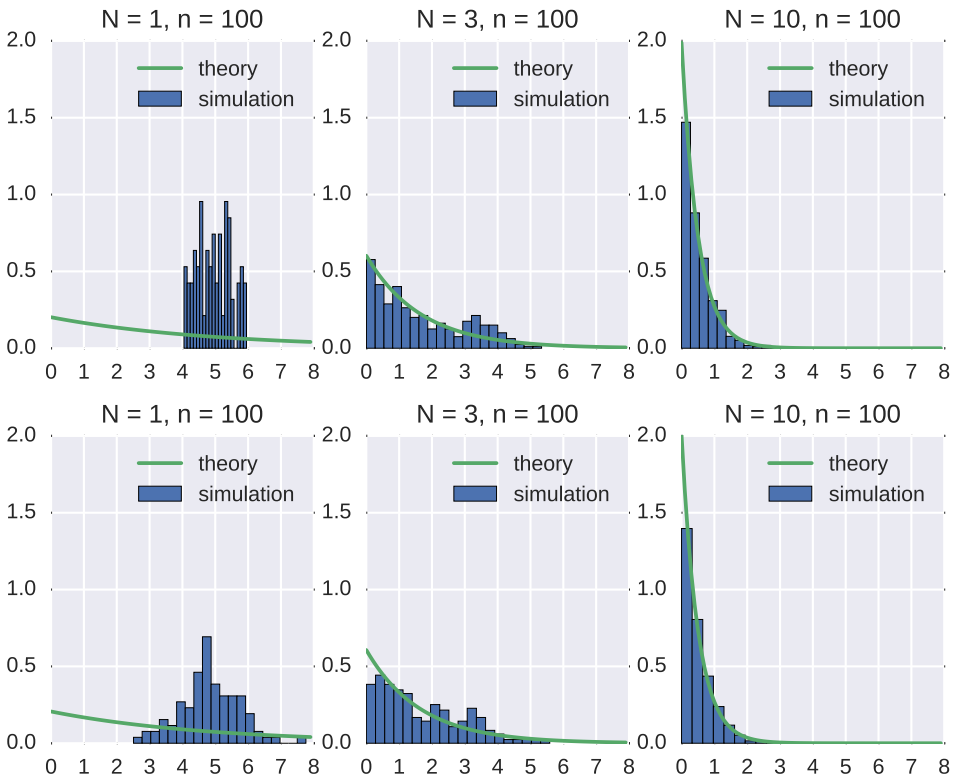


Figure 2.1: The interarrival process as seen by the shop owner. Observe that the green line intersects the y-axis at level $N/5$, which is equal to the arrival rate when N persons visit the shop. The parameter L is the simulation length, i.e., the number of arrivals per customer.

have been waiting for the next arrival to occur, the time that it occurs in the next h seconds remains the same. This property seems to be vindicated also in practice: suppose that a patient with a broken arm just arrived at the emergency room of a hospital, what does that tell us about the time the next patient will be brought in? Not much, as most of us will agree.

It can be shown that only exponential random variables have the memoryless property. The proof of this fact requires quite some work; we refer the reader to the literature if s/he want to check this.

Finally, the reader should realize that it is simple, by means of computers, to generate exponentially distributed inter-arrival times. Thus, it is easy to use such exponentially distributed random variables to simulate queueing systems.

Exercise 2.1.1 (Conditional probability)

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

Exercise 2.1.2 Assume that the time X to fail of a machine is uniformly distributed on the interval $[0,10]$. If the machine fails at time t, the cost to repair it is $h(t)$. What is the expected repair cost?

Exercise 2.1.3 Show that an exponentially distributed random variable is memoryless.

Exercise 2.1.4 If the random variable $X \sim \exp(\lambda)$, show that

$$\mathbb{E}[X] = \int_0^\infty t \, \mathrm{d}F(t) = \int_0^\infty t f(t) \, \mathrm{d}t = \int_0^\infty t\lambda e^{-\lambda t} \, \mathrm{d}t = \frac{1}{\lambda},$$

where f is the density function of the distribution function F of X.

Exercise 2.1.5 If the random variable $X \sim \exp(\lambda)$, show that

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

Exercise 2.1.6 If the random variable $X \sim \exp(\lambda)$, show that the *variance*

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

Recall in particular this middle term to compute $\mathbb{V}[X]$; it is very practical.

Exercise 2.1.7 Define the *square coefficient of variation (SCV)* as

$$C_a^2 = \frac{\mathbb{V}[X]}{(\mathbb{E}[X])^2}. \tag{2.1}$$

Prove that when X is exponentially distributed, $C_a^2 = 1$. As will become clear later, the SCV is a very important concept in queueing theory. Memorize it as a measure of *relative variability*.

Exercise 2.1.8 If X is an exponentially distributed random variable with parameter λ, show that its moment generating function

$$M_X(t) = \mathbb{E}\left[e^{tX}\right] = \frac{\lambda}{\lambda - t}.$$

Exercise 2.1.9 Let A_i be the arrival time of customer i and set $A_0 = 0$. Assume that the inter-arrival times $\{X_i\}$ are i.i.d. with exponential distribution with mean $1/\lambda$ for some $\lambda > 0$. Prove that the density of $A_i = X_1 + X_2 + \cdots + X_i = \sum_{k=1}^i X_k$ with $i \geq 1$ is

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

Exercise 2.1.10 Assume that the inter-arrival times $\{X_i\}$ are i.i.d. and $X_i \sim \exp(\lambda)$. Let $A_i = X_1 + X_2 + \cdots + X_i = \sum_{k=1}^i X_k$ with $i \geq 1$. Use the density of A_i or the moment generating function of A_i to show that

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

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

Exercise 2.1.11 If $X \sim \exp(\lambda)$ and $Y \sim \exp(\mu)$ and X and Y are independent, show that

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

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

Exercise 2.1.12 If $A \sim \exp(\lambda)$, $S \sim \exp(\mu)$ and independent, show that

$$\mathbb{P}\{A \leq S\} = \frac{\lambda}{\lambda + \mu}.$$

Exercise 2.1.13 A machine serves two types of jobs. The processing time of jobs of type i, $i = 1, 2$, is exponentially distributed with parameter μ_i. The type T of job is random and independent of anything else, and such that $\mathbb{P}\{T = 1\} = p = 1 - q = 1 - \mathbb{P}\{T = 2\}$. (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.) What is the expected processing time and what is the variance?

Exercise 2.1.14 Try to make Figure 2.1 with simulation.

2.2 Poisson Distribution

In this section we provide a derivation of, and motivation for, the Poisson process, and clarify its relation with the exponential distribution at the end.

Consider a machine that fails occasionally. Let us write $N(s,t)$ for the number of failures occurring during a time interval of $(s,t]$. We assume, without loss of generality, that repairs are instantaneous. Clearly, as we do not know in advance how often the machine will fail, we model $N(s,t)$ as a random variable for all times s and t.

Our first assumption is that the failure behavior of the machine does not significantly change over time. Then it is reasonable to assume that the expected number of failure is proportional to the length of the interval T. Thus, it is reasonable to assume that there exists some constant λ such that

$$\mathbb{E}[N(s,t)] = \lambda(t - s) \tag{2.2}$$

The constant λ is often called the *arrival rate*, or failure rate in this case.

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

To find the distribution of $N(0,t)$, let us split the interval $[0,t]$ into n sub-intervals, all of equal length, and ask: 'What is the probability that the machine will fail in some given sub-interval.' By our second assumption, the failure behavior is constant over time. Therefore, the probability p to fail in each interval should be equal. Moreover, if n is large, p must be small, for otherwise (2.2) could not be true. As a consequence, if the time intervals are very small, we can safely neglect the probability that two or more failures occur in one such tiny interval.

As a consequence, then, we can model the occurrence of a failure in some period i as a Bernoulli distributed random variable B_i such that $\mathbb{P}\{B_i = 1\}$ and $\mathbb{P}\{B_i = 0\} = 1 - \mathbb{P}\{B_i = 1\}$,

and we assume that $\{B_i\}$ are independent. The total number of failures $N_n(t)$ that occur in n intervals is then binomially distributed

$$\mathbb{P}\{N_n(t) = k\} = \binom{n}{k} p^k (1-p)^{n-k}. \tag{2.3}$$

In the exercises we ask you to use this to motivate that, in some appropriate sense, $N_n(t)$ converges to $N(t)$ for $n \to \infty$ such that

$$\mathbb{P}\{N(t) = k\} = e^{-\lambda t} \frac{(\lambda t)^k}{k!}. \tag{2.4}$$

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

Moreover, if there are no failures in some interval $[0,t]$, then it must be that $N(t) = 0$ and A_1, i.e., the occurrence of the first failure, must be larger than t. Therefore,

$$\mathbb{P}\{A_1 > t\} = \mathbb{P}\{X_1 > t\} = \mathbb{P}\{N(t) = 0\} = e^{-\lambda t} \frac{(\lambda t)^0}{0!} = e^{-\lambda t}.$$

The relations we discussed above are of paramount importance in the analysis of queueing process. We summarize this by a theorem.

Theorem 2.2.1. A counting process $\{N(t)\}$ is a Poisson process with rate λ if and only if the inter-arrival times $\{X_i\}$, i.e., the times between consecutive arrivals, are i.i.d. and $\mathbb{P}\{X_1 \le t\} = 1 - e^{-\lambda t}$. In other words, $X_i \sim \exp(\lambda) \Leftrightarrow N(t) \sim P(\lambda t)$

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

Exercise 2.2.2 What is the difference between $N_n(t)$ and $N(t)$?

Exercise 2.2.3 Show how the binomial distribution (2.3) converges to the Poisson distribution (2.4) if $n \to \infty$, $p \to 0$ such that $np = \lambda$.

Exercise 2.2.4 If the inter-arrival times are i.i.d. and exponentially distributed with mean $1/\lambda$, prove that the number $N(t)$ of arrivals during interval $[0,t]$ is Poisson distributed.

Exercise 2.2.5 Show that if $N(t) \sim P(\lambda t)$, we have for small h,

1. $\mathbb{P}\{N(h) = n \,|\, N(0) = n\} = 1 - \lambda h + o(h)$
2. $\mathbb{P}\{N(h) = n+1 \,|\, N(0) = n\} = \lambda h + o(h)$
3. $\mathbb{P}\{N(h) \ge n+2 \,|\, N(0) = n\} = o(h)$,

where $o(h)$ is a function $f(h)$ such $f(h)/h \to 0$ as $h \to 0$.

Exercise 2.2.6 Assume a timer fires at times $0 = T_0 < T_1 < T_2 < \cdots$, such that $T_k - T_{k-1} \sim \exp(\lambda)$. Define $N(t) = \sum_{k=0}^{\infty} k \, \mathbb{1}_{T_k \le t < T_{k+1}}$, where $\mathbb{1}$ is the *indicator function*, that is, $\mathbb{1}_A = 1$ if the event A is true, and $\mathbb{1}_A = 0$ if A is not true. What is the distribution of $N(t)$?

Exercise 2.2.7 (Merged Poisson streams form a new Poisson process with the sum of the rates.) Assume that $N_a(t) \sim \mathrm{P}(\lambda t)$, $N_s(t) \sim \mathrm{P}(\mu t)$ and independent. Show that $N_a(t) + N_s(t) \sim \mathrm{P}((\lambda+\mu)t)$.

Exercise 2.2.8 Assume that $N_a(t) \sim \mathrm{P}(\lambda t)$, $N_s(t) \sim \mathrm{P}(\mu t)$ and independent. Use that $N_a(t) + N_s(t) \sim \mathrm{P}((\lambda+\mu)t)$ to conclude

$$\mathbb{P}\{N_a(h) = 1, N_s(h) = 0 | N_a(h) + N_s(h) = 1\} = \frac{\lambda}{\lambda+\mu}.$$

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

Exercise 2.2.9 Assume that $N_a(t) \sim \mathrm{P}(\lambda t)$, $N_s(t) \sim \mathrm{P}(\mu t)$ and independent. What is actually the meaning of the event $\{N_a(h) = 1, N_s(h) = 0\} \cap \{N_a(h) + N_s(h) = 1\}$?

Exercise 2.2.10 (A Bernoulli-thinned Poisson process is still a Poisson process) Consider a Poisson process. Split the process in the following way. When a job arrives, throw a coin that lands heads with probability p and tails with $q = 1 - p$. When the coin lands heads, call the job of type 1, otherwise of type 2. Another way of thinning is by modeling the stream of people passing a shop as a Poisson process with rate λ. With probability p a person decides, independent of anything else, to enter the shop.

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

Exercise 2.2.11 Show that $\mathbb{E}[N(t)] = \lambda t$ and $\mathbb{V}[N(t)] = \lambda t$. Why is the SCV of $N(t)$ equal to $1/\lambda t$? Conclude that the relative variability of $N(t)$ becomes smaller as t becomes larger. Observe that the SCV of a Poisson distributed random variable is not the same as the SCV of an exponentially distributed random variable.

2.3 Kendall's Notation to Characterize Queueing Processes

As will become apparent in Sections 2.4 and 2.5, the construction of any queueing process involves three main elements: the distribution of the inter-arrival times between consecutive jobs, the distribution of the service times of the individual jobs, and the number of servers present to process jobs. In this characterization it is implicit that the inter-arrival times form a set of i.i.d. (independent and identically distributed) random variables, the service times are also i.i.d., and finally, the interarrival times and service times are mutually independent.

To characterize the type of queueing process it is common to use the *abbreviation* $A/B/c/K$ where A is the distribution of the interarrival times, B the distribution of the services, c the number of servers, and K the size of the queue. In this notation it is assumed that jobs are served in first-in-first-out (FIFO) order; FIFO scheduling is also often called first-come-first-serve (FCFS).

Let us illustrate the shorthand $A/B/c/K$ with some examples:

- $M/M/1$: the distribution of the interarrival times is *M*emory-less, hence exponential, the service times are also *M*emoryless, and there is 1 server. As K is unspecified, it is assumed to be infinite.
- $M/M/c$: A *multi-server* queue with c servers in which all servers have the same capacity.

Jobs arrive according to a Poisson process and have exponentially distributed processing times.

- $M(n)/M(n)/1$: the interarrival times are exponential, just as the service times, but the rates of the arrival and service processes may depend on the queue length n.
- $M/M/c/K$: interarrival times and process times are exponential, and the *system capacity* is K jobs. Thus, the queue can contain at most $K-c$ jobs.[3]
- $M^X/M/1$: Customers arrive with exponentially distributed interarrival times. However, each customer brings in a number of jobs, known as a batch. The number of jobs in each batch is distributed as the random variable X. Thus, the arrival process of work is *compound Poisson*.
- $M/G/1$: the interarrival times are exponentially distributed, the service times can have any *G*eneral distribution (with finite mean), and there is 1 server.
- $M/G/\infty$: exponential interarrival times, service times can have any distribution, and there is an unlimited supply of servers. This is also known as an *ample* server. Observe that in this queueing process, jobs actually never have wait in queue; upon arrival there is always a free server available.
- $M/D/1-LIFO$. Now job service times are *D*eterministic, and the service sequence is last-in-first-out (LIFO).
- $G/G/1$: generally distributed interarrival and service times, 1 server.

In the sequel we will use Kendall's notation frequently to distinguish the different queueing models. Ensure that you familiarize yourself with this notation.

Exercise 2.3.1 What are some advantages and disadvantages of using the Shortest Processing Time First (SPTF) rule to serve jobs? (Look up the definition on wikipedia or Hall [1991])

2.4 Construction of Discrete-Time Queueing Processes

In this section we discuss a case as this provides real-life motivation to analyze queueing systems. After decribing the case, we we develop a set of recursions by which we can construct queueing systems in discrete time. As it turns out, this case is too hard to analyze by mathematical means, so that simulation is the best way forward. Interestingly, the structure of the simulation is very simple. As such, simulation is an exceedingly convincing tool to communicate the results of an analysis of a queueing system to managers (and the like).

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

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

[3] Sometimes, the K stands for the capacity in the queue, not the entire system. The notation differs among authors. Due to the lack of consistency, I might also not be consistent; be warned.

1. Not all psychiatrists have the same amount of time available per week to do intakes. This is not a problem during weeks that all are present. However, psychiatrists tend to take holidays, visit conferences, and so on. So, if the psychiatrist with the most intakes per week would go on leave, this might affect the behavior of the queue length considerably. This raises the question about the difference in allocation of capacity allotted to the psychiatrists. What are the consequences on the distribution and average of the waiting times if they would all have the same weekly capacity?

2. The psychiatrists tend to plan their holidays after each other, to reduce the variation in the service capacity. What if they would synchronize their holidays, to the extent possible, rather than spread their holidays?

3. Finally, suppose the psychiatrists would do 2 more intakes per week in busy times and 2 less in quiet weeks. Assuming that the system is stable, i.e., the service capacity exceeds the demand, then on average the psychiatrists would not do more intakes, i.e., their workload would not increase, but the queue length may be controlled better.

To evaluate the effect of these suggestions on reducing the queueing dynamics we develop a simple simulator and a number of plots. The simulation of a queueing system involves the specification of its behavior over time. The easiest method to construct queueing processes, hence to develop simulations, is to 'chop up' time in periods and develop recursions for the behavior of the queue from period to period. Note that the length of such a period depends on the case for which the model is developed. For instance, to study queueing processes at a supermarket, a period can consist of 5 minutes, while for a production environment, e.g., a job shop, it can be a day, or even a week.

Using fixed sized periods has it advantages as it does not require 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 need to be specified, which is useful since in many practical settings, e.g., production environments, it is easier to provide data in these terms then in terms of inter-arrival and service times. It is, however, necessary to make some careful choices about timing.

Let us *define*

$$\begin{aligned} a_k &= \text{number of jobs that arrive at period } k, \\ c_k &= \text{number of jobs that can be served during period } k, \\ d_k &= \text{number of jobs that depart the queue } \textit{during} \text{ period } k, \\ Q_k &= \text{number of jobs in queue at the } \textit{end} \text{ of period } k. \end{aligned} \tag{2.5}$$

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

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

$$d_k = \min\{Q_{k-1}, c_k\}, \tag{2.6a}$$

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

$$Q_k = Q_{k-1} - d_k + a_k. \tag{2.6b}$$

As an example, suppose that $c_k = 7$ for all k, and $a_1 = 5$, $a_2 = 4$ and $a_3 = 9$; also $Q_0 = 8$. Then, $d_1 = 7$, $Q_1 = 8-7+5 = 6$, $d_2 = 6$, $Q_2 = 6-6+4 = 4$, $d_3 = 4$, $Q_3 = 4-4+9 = 9$, and so on.

Of course we are not going to carry out these computations by hand. Typically we use company data about the sequence of arrivals $\{a_k\}_{k=1,2,\ldots}$ and the capacity $\{c_k\}_{k=1,\ldots}$ and feed this data into a computer to compute the recursions (2.6). If we do not have sufficient data we make a probability model for these data and use the computer to generate random numbers with, hopefully, similar characteristics as the real data. At any rate, from this point on we assume that it is easy, by means of computurs, to obtain numbers $a_1, \ldots, a_n$ for $n \gg 1000$, and so on.

We now show how to adapt (2.5) and (2.6) to the case introduced above.

As a first step we model the arrival process of patients as a Poisson process, c.f., 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 slighly less than 12 per week; in the simulation we set it to $\lambda = 11.8$ per week. We model the capacity in the form of a matrix such that row i corresponds to the weekly capacity of psychiatrist i:

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

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

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

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

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

$$C = \begin{pmatrix} 0 & 2 & 2 & 2 & 2 & 0 & \ldots \\ 2 & 0 & 2 & 2 & 2 & 2 & \ldots \\ 3 & 3 & 0 & 3 & 3 & 3 & \ldots \\ 4 & 4 & 4 & 0 & 4 & 4 & \ldots \\ 4 & 4 & 4 & 4 & 0 & 4 & \ldots \end{pmatrix},$$

Hence, the total average capacity must be $4/5 \cdot (2+2+2+4+4) = 12$ 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 (2.6) to simulate the queue length process for the four different scenarios proposed by the psychiatrists, unbalanced versus balanced capacity, and spread out holidays versus simultaneous holidays. The results are shown in Figure 2.2. It is apparent that suggestions 1 and 2 above do not significantly affect the behavior of the queue length process.

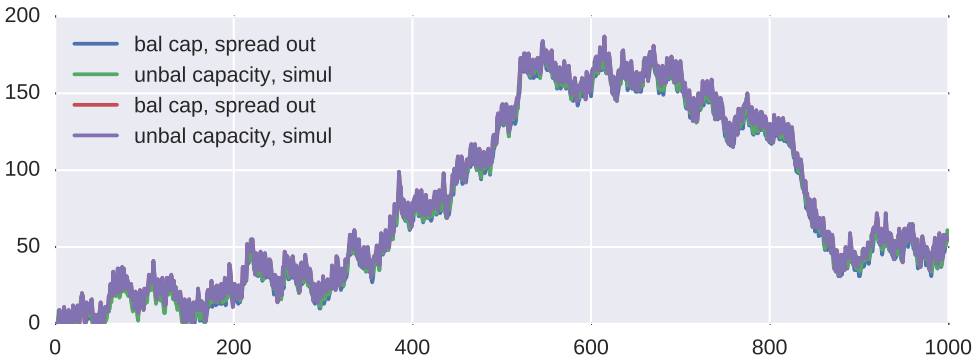


Figure 2.2: Effect of capacity and holiday plans

Now we consider suggestion 3, which comes down to doing more intakes when it is busy, and do less when it is quiet. A simple rule to implement this is by considering last week's queue Q_{n-1}: if $Q_{n-1} < 12$, i.e., the service capacity of one week, then do e intakes less. Here, $e = 1$ or 2, or perhaps a larger number; it corresponds to the amount of control we want to exercise. When $Q_{n-1} > 24$, i.e., larger than two weeks of intakes, do e intakes more. Let's consider three different control levels, $e = 1$, $e = 2$, and $e = 5$; thus in the last case all psychiatrists do one extra intake. The previous simulation shows that it is safe to disregard the holiday plans, so just assume a flat service capacity of 12 intakes a week.

Figure 2.3 shows a striking difference indeed. The queue does not explode any more, and already taking $e = 1$ has a large influence.

Figure 2.3: Controling the number of intakes

From the simulation experiment we learn that changing holiday plans or spreading the work

over multiple servers, i.e., psychiatrists, does not significantly affect the queueing behavior. However, controlling the service rate as a function of the queue length improves the situation quite dramatically.

Thus, we made models for the arrival and service processes and then developed a simple recursion of the type (2.6) to *construct* a queueing process. Finally, for the analysis, we made plots of these processes.

In more abstract terms the study of queueing system is focussed on studying the probabilistic properties of the queueing length process and related concepts such as waiting time, server occupancy, fraction of customers lost, and so on. Once we have constructed the queueing process we can compute all performance measures of relevance, such as the average waiting time, c.f. Section 2.8. If it turns out that the performance of the system is not according to what we desire, we can change parts of the system with the aim to improve the situation and assess the effect of this change. For instance, if the average waiting time is too long, we might add service capacity with the aim to reduce the service times, hence reduce the average waiting time. With simulation it is easy to study the effect of, hence evaluate, such decisions.

Observe that, even with these simple recursions, we can obtain considerable insight into this, otherwise, very complicated controlled queueing process. (If the reader doubts the value of simulation, s/he should try to develop other mathematical methods to analyze multi-server queueing system with vacations, of which this is an example. Warning, do not even attempt: you'll fail as it is most probably too hard.) The simplicity of the recursions is deceitful, but with these simple recursions we can analyse many practical queueing situations. Together with students the author applied it numerous times, for instance,

- Should a certain hospital invest in a new MRI scanner to reduce waiting times?
- When to switch on and off a tin bath at an eletronics component factory?
- What is the effect of reducing the number of jobs in a paint factory?
- Post parcel routing in a post sorting center.
- Controlling inventories at various companies.
- Throughput time analysis at courts.

And so on, and so on. The recursions are indeed astoningly useful. We therefore urge the reader to practice with this type of queueing modeling. The exercises below provide ample material for this purpose.

In passing we remark that yet more powerful simulations can be carried out with, so-called, event-based simulations. We refer to [Hall, 1991, Section 4.5] for a general description of this technique. The author applied this to quite complicated queueing networks:

- Analysis of the packaging process of a large beer factory.
- Performance analysis of large telecommunication networks
- Optimization of truck routing and queueing for a sugar factory.

Due to lack of time, we decided not to include this.

The reader should understand from the above case that, once we have the recursions, we can analyze the system and make plots to evaluate suggestions for improvement. Thus, getting the

recursions is crucial to construct, i.e., model, queueing processes. For this reason, most of the exercises below focus on obtaining recursions for many different queueing systems.

Exercise 2.4.1 What are the consequences of setting $d_k = \min\{Q_{k-1} + a_k, c_k\}$ rather than the definition (2.6a)?

Exercise 2.4.2 (Queue with Blocking) Consider a queueing system under daily review, i.e., at the end of the day the queue length is measured. We assume that at the end of the day no jobs are still in service. We assume that jobs that arrive at day n cannot be served in day n. The queue length cannot exceed level K. Formulate a set of recursions to cover this case.

Exercise 2.4.3 (Yield loss) A machine produces items, but part of the items turn out to be faulty, and have to be made anew. Make assumptions about the failure process, and develop a set of recursions to cover this case.

Exercise 2.4.4 (Rework) A machine produces items, but part of the items do not meet the quality requirements after the first service but need some extra service time but less than an entirely new arriving job. Make a model to analyze this case. Compare this case with the yield loss problem above.

Exercise 2.4.5 (Cost models) A single-server queueing station processes customers. At the start of a period the server capacity is chosen, so that for period k the capacity is c_k. Demand that arrives in a period can be served in that period. It costs β per unit time per unit processing capacity to operate the machine, i.e., to have it switched on. There is also a cost h per unit time per job in the system. Make a cost model to analyze the long-run average cost for this case.

Exercise 2.4.6 (N-policies) A machine can switch on and off. If the queue length hits N, the machine switches on, and if the system becomes empty, the machine switches off. It costs K to switch on the machine. There is also a cost β per unit time while the machine is switched on, and it costs h per unit time per customer in the system. Make a cost model.

Exercise 2.4.7 (Queue with setups)[use=false] One server serves two parallel queues, one at a time. After serving a batch of 20 jobs of one queue the server moves to the other queue. The change of queue requires one period setup time.

Exercise 2.4.8 How would you model (in terms of recursions) a server whose capacity depends on the queue length?

Exercise 2.4.9 (Fair queueing) One server serves two queues. Each queue receives service capacity in proportion to its queue length. Derive a set of recursions to analyze this situation.

Exercise 2.4.10 (Priority queuing) Another interesting situation is a system with two queues served by one server, but such that one queue gets non-preemptive priority over the other queue. Again find a set of recursions to describe this case.

Exercise 2.4.11 (Queues with reserved service capacity) Consider a single-server that serves two parallel queues. Each queue receives a minimal service capacity every period. Reserved capacity unused for one queue can be used to serve the other queue. Any extra capacity beyond the reserved capacity is given to queue A with priority. Formulate a set of recursions to analyze this situation.

Exercise 2.4.12 (Queue with protected service capacity, lost capacity) Consider a single-server that serves two parallel queues. Each queue receives a minimal service capacity every period. Reserved capacity unused for one queue cannot be used to serve the other queue. Any extra capacity beyond the reserved capacity is given to queue A with priority. Formulate a set of recursions to analyze this situation.

Exercise 2.4.13 (Tandem networks) Consider a production network with two production stations in tandem, that is, the jobs processed by station A are in the next period to the downstream Station B. Extend the recursions of (2.6) to simulate this situation.

Exercise 2.4.14 (A tandem queue with blocking) Consider a production network with two production stations in tandem with blocking: when intermediate queue, i.e., the queue front of Station B, exceeds some level M, then station A has to stop producing, and when $Q_k^B < M$ station A is not allowed to produce more then the intermediate queue can contain. Extend the recursions of (2.6) to simulate this situation.

Exercise 2.4.15 (Merging departure streams) Consider another production situation with two machines, A and B say, that send their products to Station C. Derive a set of recursion relations to simulate this system.

Exercise 2.4.16 (Merging incoming streams) Consider a single-server queue that servers two customer 'streams' in a FIFO discipline. Thus, both streams enter one queue that is served by the server. Let $\{a_k^a\}$ be the number of arrivals of stream a in period k and $\{a_k^b\}$ be the number of arrivals of stream b. Find a set of recursions by which it becomes possible to analyze the waiting time distribution of each of the streams.

Exercise 2.4.17 (Splitting streams) Consider a machine (a paint mixing machine) that produces products for two separate downstream machines A and B (two different paint packaging machines), each with its own queue. Suppose we want to analyze the queue in front of station A. For this we need to know the arrivals to station A, that is, the departures of the mixing station that go to station A. Provide a set of recursions to simulate this system.

Exercise 2.4.18 (Inventory control) The recursions used in the exercises above can also be applied to analyze inventory control policies. Consider a production system that can produce maximally M_k items per week during normal working hours, and maximally N_k items during extra (weekend and evening hours). Let, for period k,

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

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

Exercise 2.4.19 (Estimating the lead time distribution.) Take $d_k = \min\{Q_{k-1} + a_k, c_k\}$, and assume that jobs are served in FIFO sequence.

1. Find an expression for the earliest possible departure time $l_-(k)$ of a job that arrives at time k.
2. Find an expression for the latest possible departure time $l_+(k)$ of a job that arrives at time k.
3. How can you use l_- and l_+ to bound the waiting times?

Exercise 2.4.20 Implement Eqs. 2.6 in a computer program and simulate a simple single-server queueing system.

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

In the previous section we considered time in discrete 'chunks', minutes, hours, days, and so on. For given numbers of arrivals and capacity per period we use a set of recursions (2.6) to compute the departures and queue length per period. Another way to construct a queueing system is to consider inter-arrival times between consecutive customers and the service times each of these customers require. With this we obtain a description of the queueing system in continuous time. The goal of this section is to develop a set of recursions for the $G/G/1$ single-server queue.

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

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

$$A_k = \min\{t : A(t) \geq k\}, \tag{2.7}$$

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

$$X_k = A_k - A_{k-1}. \tag{2.8}$$

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

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

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

$$A(t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \leq t}, \tag{2.9a}$$

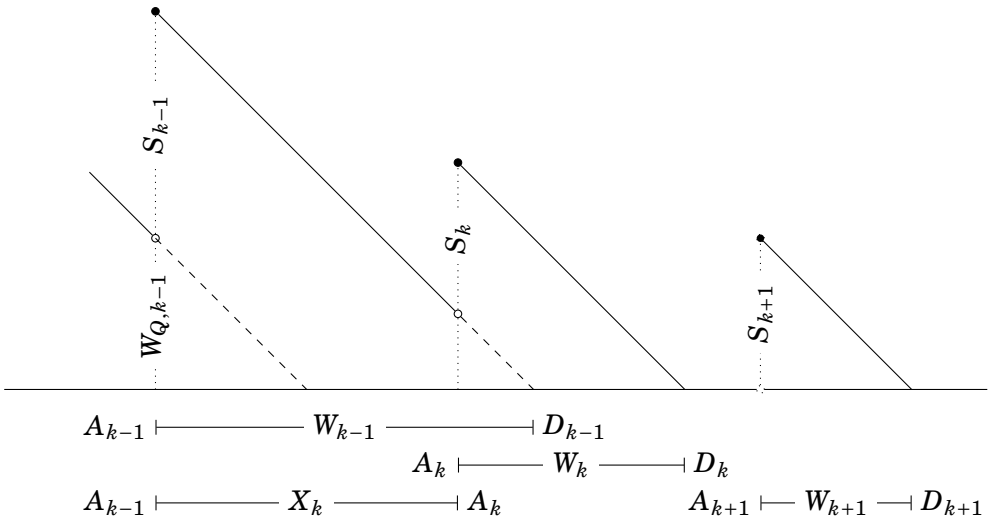


Figure 2.4: Construction of the $G/G/1$ queue in continuous time. The sojourn time W_k of the kth job is sum of the work in queue $W_{Q,k}$ at its arrival epoch A_k and its service time S_k; its departure time is then $D_k = A_k + W_k$. The waiting time of job k is clearly equal to $W_{k-1} - X_k$. We also see that job $k+1$ arrives at an empty system, hence its sojourn time $W_{k+1} = S_{k+1}$. Finally, the virtual waiting time process is shown by the lines with slope -1.

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

$$A(t) = \max\{k : A_k \le t\}. \tag{2.9b}$$

Clearly, we see that from the inter-arrival times $\{X_k\}$ it is possible to construct $\{A_k\}$ and $\{A(t)\}$, and the other way around, from $\{A(t)\}$ we can find $\{A_k\}$ and $\{X_k\}$. Figure 2.4 shows these relations graphically. To memorize it may be helpfull to write it like this:

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

To compute the departure times $\{D_k\}$ we proceed in stages. The first stage is to construct the *waiting time in queue* $\{W_{Q,k}\}$ as seen by the arrivals. In Figure 2.4 observe that the waiting time of the kth arrival must be equal to the waiting time of the $k-1$th customer plus the amount of *service time* required by job $k-1$ minus the time that elapses between the arrival of job $k-1$ and job k, unless the server becomes idle between jobs $k-1$ and k. In other words,

$$W_{Q,k} = [W_{Q,k-1} + S_{k-1} - X_k]^+, \tag{2.10}$$

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

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

$$D_{Q,k} = A_k + W_{Q,k},$$

because a job can only move to the server after its arrival plus the time it needs to wait in queue. Note that we here explicitly use the FIFO assumption.

Right after the job moves from the queue to the server, its service starts. Thus, $D_{Q,k}$ is the epoch at which the service of job k starts. After completing its service, the job leaves the system. Hence, the *departure time of the system* is

$$D_k = D_{Q,k} + S_k.$$

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

$$W_k = D_k - A_k = D_{Q,k} + S_k - A_k = W_{Q,k} + S_k, \tag{2.11}$$

where each of these equations has its own interpretation.

A bit of similar reasoning gives another recursion for W_k:

$$\begin{aligned} W_{Q,k} &= [W_{k-1} - X_k]^+, \\ W_k &= W_{Q,k-1} + S_k = [W_{k-1} - X_k]^+ + S_k. \end{aligned} \tag{2.12}$$

from which follows a recursion for D_k

$$D_k = A_k + W_k. \tag{2.13}$$

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

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

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

$$L(t) = A(t) - D(t) + L(0), \tag{2.14}$$

where $L(0)$ is the number of jobs in the system at time $t = 0$; typically we assume that $L(0) = 0$. Thus, if we were to plot $A(t)$ and $D(t)$ as functions of t, then the difference $L(t)$ between the graphs of $A(t)$ and $D(t)$ tracks the number in the system, see Figure 2.5.

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

$$L_Q(t) = A(t) - D_Q(t)$$

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

$$L_s(t) = L(t) - L_Q(t).$$

Finally, the *virtual waiting time process* $\{V(t)\}$ is the amount of waiting that an arrival would see if it would arrive at time t. To construct $\{V(t)\}$, we simply draw lines that start at points (A_k, W_k) and have slope -1, unless the line hits the x-axis, in which case the virtual waiting time remains zero until the next arrival occurs. Thus, the lines with slope -1 in Figure 2.4 show (a sample path of) the virtual waiting time.

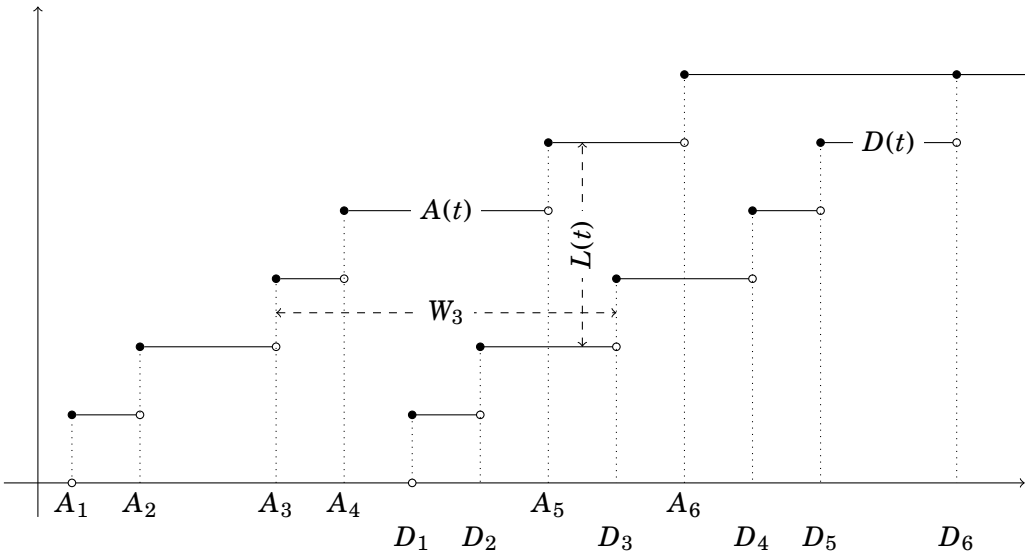


Figure 2.5: Relation between the arrival process $\{A(t)\}$, the departure process $\{D(t)\}$, the number in the system $\{L(t)\}$ and the waiting times $\{W_k\}$.

Observe that, just as in Section 2.4, we have obtained a set of recursions by which we can run a simulation of a queueing process of whatever length we need, provided we have a sequence of have inter-arrival times $\{X_k\}$ and service times $\{S_k\}$. A bit of experimentation with computer programs such as *R* or python will reveal that this is easy.

Exercise 2.5.1

1. 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.

Exercise 2.5.2 Suppose that $X_k \in \{1,3\}$ such that $\mathbb{P}\{X_k = 1\} = \mathbb{P}\{X_k = 3\}$ and $S_k \in \{1,2\}$ with $\mathbb{P}\{S_k = 1\} = \mathbb{P}\{S_k = 2\}$. If $W_{Q,0} = 3$, what are the distributions of $W_{Q,1}$ and $W_{Q,2}$?

Exercise 2.5.3 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 $\mathbb{P}\{S - X \le x\}$.

Exercise 2.5.4 What are the meanings of $A_{A(t)}$ and $A(A_n)$?

Exercise 2.5.5 Would it make sense to define $A(t) = \min\{k : A_k \ge t\}$?

Exercise 2.5.6 Define $L_Q(t)$ as the number of job in queue, and $L_s(t)$ as the number of jobs in service. Likewise, let $D_Q(t)$ be the number of jobs that departed from the queue up to time t.

1. Why don't we need separate notation for $D_s(t)$, the number of jobs that departed from the server?
2. Is $D_Q(t) \le D(t)$ or $D_Q(t) \ge D(t)$?
3. Why is $L(t) = L_Q(t) + L_s(t)$?

4. Express $L_Q(t)$ and $L_s(t)$ in terms of $A(t)$, $D_Q(t)$ and $D(t)$.

5. Consider a multi-server queue with m servers. Suppose that at some t it happens that $D_Q(t) - D_s(t) < m$ even though $A(t) - D_s(t) > m$. How can this occur?

Exercise 2.5.7 Show that $L(t) = A(t) - D(t)$ implies that

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

Show also that if $L(A_k) > 0$, i.e., the system contains at least one job at the time of the kth arrival, then $A_k \le D_{k-1}$, i.e., job k arrives before job $k-1$ departs.

Exercise 2.5.8 Can you derive following (algorithmic efficient) procedure to compute the number of jobs in the system as seen by arrivals,

$$L_k = L_{k-1} + 1 - \sum_{i=k-1-L_{k-1}}^{k-1} 1\{D_i < A_k\}?$$

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

Exercise 2.5.9 Provide a specification of the virtual waiting time process $\{V(t)\}$ for all t.

Exercise 2.5.10 Another set of recursions to compute the arrival and departure times for a given set of inter-arrival and service times is the following:

$$\begin{aligned} A_k &= A_{k-1} + X_k, \\ D_k &= \max\{A_k, D_{k-1}\} + S_k. \end{aligned} \tag{2.15}$$

Now the computation of the waiting times is trivial: $W_k = D_k - A_k$.

1. Why do the recursions Eq. (2.15) work?

2. (Very difficult) Extend the above recursions to a situation in which one queue is served by two servers.

3. (Impossible?) Extend the above recursions to a situation in which one queue is served by $m > 2$ servers.

Exercise 2.5.11 Implement the above recursions in excel or some other computer program such as R, python, or julia, and check the results of the previous exercise.

Exercise 2.5.12 (Multiple queues, for the interested) Suppose one server serves two queues, such that jobs in queue A are served with priority over jobs in queue B. Assume that service is not-preemptive. Can you develop a similar set of recursions for each of the queues, or, otherwise, an algorithm?

Exercise 2.5.13 (LIFO queue) Can you develop a set of recursions or algorithm for a LIFO (Last-In-First-Out) queue?

2.6 Queueing Processes as Regulated Random Walks

In the construction of queueing processes as set out in section 2.4 we are given two sequences of i.i.d. random variables: the number of arrivals $\{a_k\}$ per period and the service capacities $\{c_k\}$. Assuming that jobs have to wait at least one period before they can be processed, the departure and queue length processes are generated by the recursions

$$\begin{aligned} Q_k &= [Q_{k-1} + a_k - c_k]^+, \\ d_k &= Q_k + a_k - Q_{k-1}, \end{aligned} \tag{2.16}$$

where $[a]^+ = \max\{a, 0\}$. Observe now that the relation for Q_k shares a resemblance to a random walk $\{Z_k, k = 0, 1, \ldots\}$ with Z_k given by

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

Clearly, $\{Z_k\}$ is 'free', i.e., it can take positive and negative values, but $\{Q_k\}$ is restricted to the non-negative integers. In this section we show how to build the queueing process $\{Q_k\}$ from the random walk $\{Z_k\}$ by a device called a *reflection map*. Besides the elegance of this construction, the construction leads to spectacularly fast simulations. Moreover, we can use the probabilistic tools that have been developed for the random walk to analyze queueing systems. On example is the distribution of the time until an especially large queue is reached; these times can be formulated as *hitting times* of the random walk. Another example is the average time it takes to clear a large queue.

In an exercise below we derive that Q_k satisfies the relation

$$Q_k = Z_k - \min_{0 \le i \le k} Z_i \wedge 0, \tag{2.17}$$

where Z_k is defined by the above random walk and $a \wedge b = \min\{a, b\}$ [4]. This recursion leads to really interesting graphs, see Figures 2.6 and 2.7. We consider two examples. Let $a_k \sim B(0.3)$, i.e., a_k is Bernoulli-distributed with success parameter $p = 0.3$, i.e., $\mathbb{P}\{a_k = 1\} = 0.3 = 1 - \mathbb{P}\{a_k = 0\}$, and $c_k \sim B(0.4)$. As a second example, we take $a_k \sim B(0.49)$ and construct the random walk as

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

Thus, if $a_k = 1$, the random walk increases by one step, while if $a_k = 0$, the random walk decreases by one step.

Now that we have seen that random walks can be converted into queueing systems, we study in an exercise below the transient distribution, i.e., the distribution as a function of time, of the random walk $\{Z_k\}$. The aim of this exercise is to show that there is no simple function by which we can compute the transient distribution of this relatively simple random walk. Since a queueing process is typically a more complicated object (as it is a constrained random walk), our hopes to finding anything simple for the transient analysis of the $M/M/1$ queue should not be too high. And the $M/M/1$ is but the simplest queueing system; other queueing systems will be more complicated yet. We therefore give up the analysis of such transient queueing systems and we henceforth contend ourselves with the analysis of queueing systems in the limit as $t \to \infty$. This of course warrants two questions: what type of limit is actually meant here,

[4] For further detail, refer to Baccelli and Massey [1988].

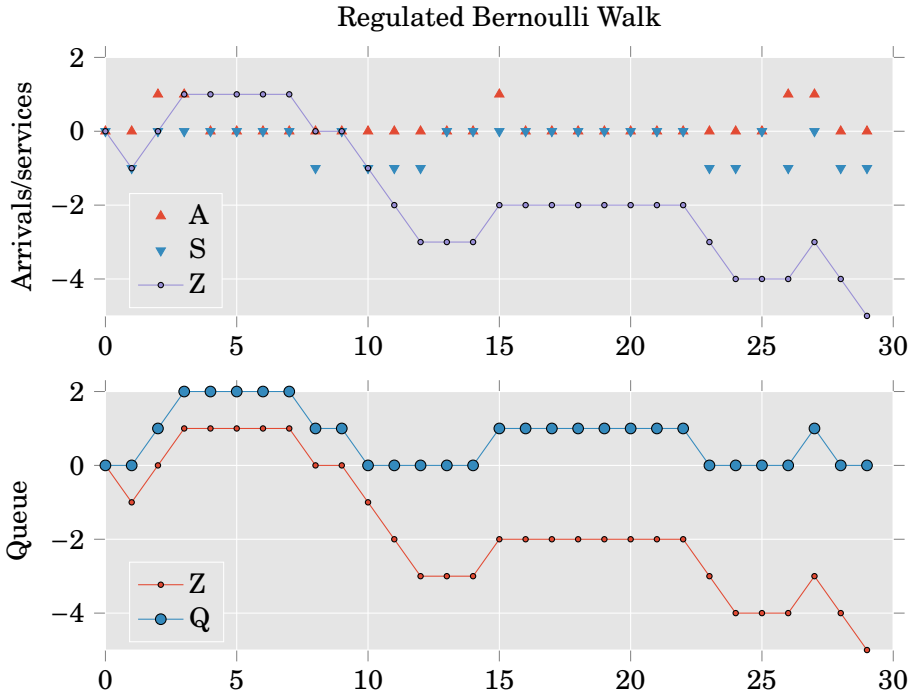


Figure 2.6: The upper panel shows a graph of the random walk Z. An upward pointing triangle corresponds to an arrival, a downward triangle to a potential service. The lower panel shows the queueing process $\{Q_k\}$ as a random walk with reflection.

and is such limiting situation reached in any reasonable time? We address these questions subsequently.

The *long-run limiting behavior* of a queueing system is an important topic by itself. The underlying question is what happens if we simulate the system for a long time. For instance, does there exist a random variable Q such that $Q_k \to Q$ in some sense? The answer to this question is in the affimative, provided some simple stability conditions are satisfied, see Section 2.7. However, it requires a considerable amount of mathematics to make this procedure precise. To sketch what has to be done, first, we need to define $\{Q_k\}$ as random variables in their own right. Note that up to now we just considered each Q_k as a *number*, i.e., a measurement or simulation of the queue length time of the kth period. Defining Q_k as a random variable is not as simple as the definition of, for instance, the number of arrivals $\{a_k\}$; these random variables can be safely *assumed* to be i.i.d. However, the queue lengths $\{Q_k\}$ are certainly not i.i.d., but, as should be apparent from Eq. (2.12), they are *constructed* in terms of recursions. Next, based on these recursions, we need to show that the sequence of distribution functions $\{G_k\}$ associated with the random variables $\{Q_k\}$ converges to some limiting distribution function G, say. Finally, it is necessary to show that it is possible to construct a random variable Q that has G as its distribution function. In this sense, then, we can say that $Q_k \to Q$. The random variable Q is known as the *steady-state limit* of the sequence of random variables $\{Q_k\}$, and the distribution G of Q is known as the *limiting* or *stationary distribution* of $\{Q_k\}$.

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

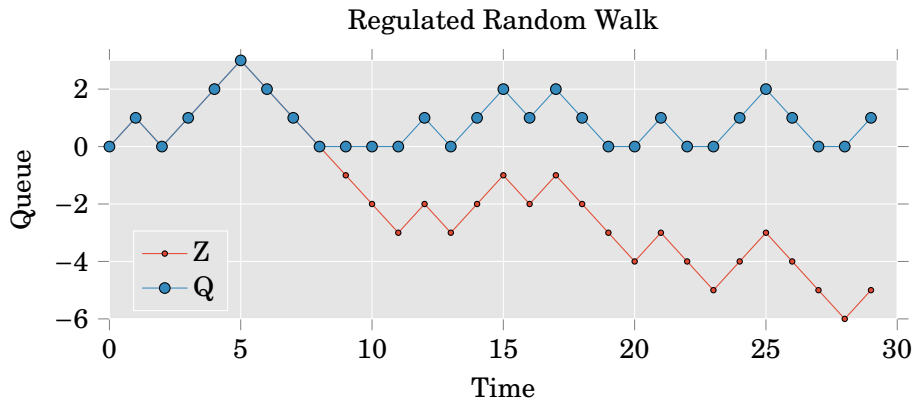


Figure 2.7: Another example of a reflected random walk.

Next, as an example, we address the rate of convergence of the sequence of waiting times $\{W_{Q,k}\}$ to a limiting random variable W_Q, where $W_{Q,k}$ is constructed according to the recursion Eq. (2.10). Suppose that $X_k \sim U\{1,2,4\}$ and $S_k \sim U\{1,2,3\}$. Starting with $W_{Q,0} = 5$ we use Eq. (2.10) to compute the *exact* distribution of $W_{Q,k}$ for $k = 1,2,\ldots,20$, c.f., the left panel in Figure 2.8. We see that when $k = 5$, the 'hump' of $\mathbb{P}\{W_{Q,5} = x\}$ around $x = 5$ is due the starting value of $W_{Q,0} = 5$. However, for $k > 10$ the distribution of $W_{Q,k}$ hardly changes, at least not visually. Apparently, the convergence of the sequence of distributions of $W_{Q,k}$ is rather fast. In the middle panel we show the results of a set of *simulations* for increasing simulation length, up to $N = 1000$ samples. Here the *emperical distribution* for the simulation is defined as

$$\mathbb{P}\{W_Q \le x\} = n^{-1} \sum_{k=1}^{n} \mathbb{1}_{W_{Q,k} \le x},$$

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

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

Exercise 2.6.1 What is the difference between the recursive schemes of Eq. (2.16) and Eq. (2.6)? Explain in the former scheme the formula for d_k.

Exercise 2.6.2 Show that

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

where Z_k is defined by the above random walk.

Exercise 2.6.3 (Transient behavior of the free $M/M/1$ queue.) The free $M/M/1$ queue is a queueing process that can become negative. Thus, it is just the random walk

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

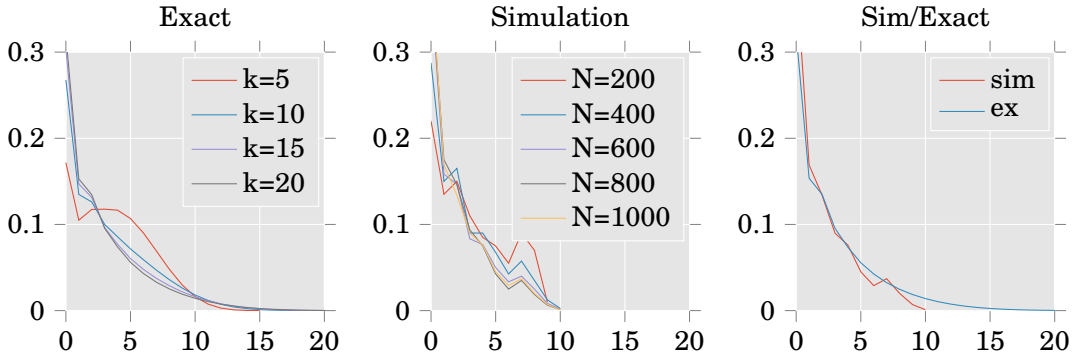


Figure 2.8: The density of $W_{Q,k}$ for $k = 5, 10, 15, 20$ computed by an exact method as compared the density obtained by simulation of different run lengths $N = 200, 400, \ldots 1000$. The right panel compares the exact density of $W_{Q,20}$ to the density obtained by simulation for $N = 1000$.

where the arrival process is a Poisson process $N_\lambda(t)$ and the departure process is a Poisson process $N_\mu(t)$. Show that

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

where $Z(0) = m$.

Now this summation is known as a modified Bessel function, but this does not make us, in some sense, much wiser. In fact, we use this result as a deterrent to study transient aspects of queueing system.

Exercise 2.6.4 Can you relate the construction of $\{Q_k\}$ as a discrete-time random walk to a continuous-time construction of the $M/M/1$ queue?

Exercise 2.6.5 Suppose that $X_k \in \{1, 3\}$ such that $\mathbb{P}\{X_k = 1\} = \mathbb{P}\{X_k = 3\}$ and $S_k \in \{1, 2\}$ with $\mathbb{P}\{S_k = 1\} = \mathbb{P}\{S_k = 2\}$. Write a computer program to see how fast the distributions of $W_{Q,k}$ converge to a limiting distribution funcion.

Exercise 2.6.6 Validate the results of Figure 2.8 with simulation.

2.7 Rate Stability and Utilization

In the analysis of any queueing process the first step should be to check the relations between the arrival, service and departure rates. The concept of rate is crucial because it captures our intuition that when, on the long run, jobs arrive faster than they can leave, the system must 'explode'. Thus, the first performance measures we need to estimate when analyzing a queueing system are the arrival and departure rate, and then we need to check that the arrival rate is smaller than the departure rate. In particular, the load, defined as the ratio of the arrival rate and service rate is of importance. In this section we define and relate these concepts. As a reminder, we keep the discussion in these notes mostly at an intuitive level, and refer to El-Taha and Stidham Jr. [1998] for proofs and further background.

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

$$\lambda = \lim_{t\to\infty} \frac{A(t)}{t}. \tag{2.18}$$

We remark in passing that this limit does not necessarily exist if $A(t)$ is some pathological function. If, however, the inter-arrival times $\{X_k\}$ are the basic data, and $\{X_k\}$ are i.i.d. and distributed as a generic random variable X with finite mean $\mathbb{E}[X]$, we can construct $\{A_k\}$ and $\{A(t)\}$ as described in Section 2.5; the strong law of large numbers guarantees that the above limit exists.

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

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

But since $A_n \to \infty$ if $n \to \infty$, it follows from Eq. (2.18) that the average inter-arrival time between two consecutive jobs is

$$\mathbb{E}[X] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} X_k = \lim_{n\to\infty} \frac{A_n}{A(A_n)} = \lim_{t\to\infty} \frac{t}{A(t)} = \frac{1}{\lambda}, \tag{2.19}$$

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

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

$$\lim_{t\to\infty} \frac{D(t)}{t} = \gamma \tag{2.20}$$

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

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

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

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

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

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

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

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

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

$$\gamma := \lim_t \frac{D(t)}{t} \le \lim_t \frac{A(t)}{t} = \lambda. \tag{2.21}$$

We call a system *(rate) stable* if

$$\lambda = \gamma,$$

in other words, the system is stable if, on the long run, jobs leave the system just as fast as they arrive. Observe that if $\lambda > \gamma$, then the queue length process $L(t) \to \infty$ as $t \to \infty$.

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

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

When $\mu \ge \lambda$ the above inequality reduces to $\gamma = \lambda$ for rate-stable systems. (It is interesting to prove this.) As it turns out, when $\mu = \lambda$ and $\mathbb{V}[S_n] > 0$ or $\mathbb{V}[X_n] > 0$ then $\lim_t L(t)/t$ does not necessarily exist. For this reason we henceforth require that $\mu > \lambda$.

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

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

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

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

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

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

Applying similar limits to the other inequality gives

$$\lambda \mathbb{E}[S] \ge \rho \ge \gamma \mathbb{E}[S].$$

Hence, if $\gamma = \lambda$, $\rho = \lambda \mathbb{E}[S]$.

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

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

Thus, the load has also the interpretation as the rate at which jobs arrive multiplied by the average amount of work per job. Finally, recall that for a system to be rate-stable, it is necessary that $\mu > \lambda$, implying in turn that $\rho < 1$. The relation $\rho = \mathbb{E}[S]/\mathbb{E}[X] < 1$ then tells us that the average time it takes to serve a job must be less than the average time between two consecutive arrivals, i.e., $\mathbb{E}[S] < \mathbb{E}[X]$.

Exercise 2.7.1 Define the departure time D_k of the kth job in terms of $\{D(t)\}$. (Use the analogy with Eq. (2.7).)

Exercise 2.7.2 For stability of the queueing process it is essential that the random variables

$$U_k = S_{k-1} - X_k$$

have negative expectation, i.e., $\mathbb{E}[U_k] = \mathbb{E}[S_{k-1} - X_k] < 0$. What is the conceptual meaning of this inequality?

Exercise 2.7.3 Let

$$U_k = S_{k-1} - X_k$$

have negative expectation, i.e., $\mathbb{E}[U_k] = \mathbb{E}[S_{k-1} - X_k] < 0$. Show that $\mathbb{E}[U_k] < 0$ implies that $\lambda < \mu$.

Exercise 2.7.4 Show that $\mathbb{E}[S]/\mathbb{E}[X]$ is the fraction of time the server is busy.

Exercise 2.7.5 Show that $\mathbb{E}[X - S]/\mathbb{E}[X]$ is the fraction of time the server is idle.

Exercise 2.7.6 If $\mathbb{E}[B]$ is the expected busy time and $\mathbb{E}[I]$ is the expected idle time, show that

$$\mathbb{E}[B] = \frac{\rho}{1-\rho}\mathbb{E}[I].$$

is the fraction of time the server is busy.

Exercise 2.7.7 Consider a queueing system with c identical servers (identical in the sense that each server has the same production rate μ). What would be a reasonable stability criterion for this system?

Exercise 2.7.8 Consider a paint factory which contains a paint mixing machine that serves two classes of jobs, A and B. The processing times of jobs of types A and B are constant and require t_A and t_B hours. The job arrival rate is λ_A for type A and λ_B for type B jobs. It takes a setup time of S_s hours to clean the mixing station when changing from paint type A to type B, and there is no time required to change from type B to A.

To keep the system (rate) stable, it is necessary to produce the jobs in batches, for otherwise the server, i.e., the mixing machine, spends a too large fraction of time on setups, so that $\mu < \lambda$. Thus, it is necessary to identify minimal batch sizes to ensure that $\mu > \lambda$. Motivate that the linear program below can be used to determine the minimal batch sizes.

$$\text{minimize } T$$

such that

$$\begin{aligned} T &= k_A t_A + S + k_B t_B, \\ \lambda_A T &< k_A, \\ \lambda_B T &< k_B. \end{aligned}$$

Exercise 2.7.9 This exercise is meant to give the reader some idea about what needs to be done to put everything on solid ground.

1. In Eq. (2.19) we replaced the limit with respect to n by a limit with respect to t. But why is this actually allowed? Use the notation $A_{A(t)}$ to show that all is OK.
2. Show that the function $t \to A(t)$ as defined by Eqs. (2.9) is right-continuous.

Exercise 2.7.10 Check with simulation that when $\lambda > \mu$ the queue length grows roughly linearly with slope $\lambda - \mu$. Thus, if $\rho > 1$, 'we are in trouble'.

2.8 (Limits of) Emperical Performance Measures

If the arrival and service processes are such that the queueing system is rate-stable, we can sensibly define other performance measures such as the average waiting time. In this section we define the second most important performance measures; the most important being the utilization ρ. We provide an overview of the relations between these performance measures in Figure 2.9.

With the construction of queueing processes in Section 2.5 we can compute the waiting time as observed by the first n, say, jobs. Thus, the average waiting time of the first n arrivals is given by $n^{-1}\sum_{k=1}^{n} W_k$. We therefore define the *expected waiting time* as

$$\mathbb{E}[W] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} W_k, \tag{2.22}$$

and the expected time in queue as

$$\mathbb{E}\left[W_Q\right] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} W_{Q,k}. \tag{2.23}$$

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

$$\mathbb{P}\{W \le x\} = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{W_k \le x}. \tag{2.24}$$

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

$$\mathbb{E}[L] = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} L_k, \tag{2.25}$$

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

$$\mathbb{P}\{L \le m\} = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{L_k \le m}. \tag{2.26}$$

A related set of performance measures follows by tracking the system's behavior over time and taking the *time-average*, rather than the average at sampling (observation) moments. Thus, if we simulate the queueing system up to time t, the *time-average number of jobs in the system* is given by

$$\mathscr{L}(t) = \frac{1}{t}\int_0^t L(s)\,\mathrm{d}s = \frac{1}{t}\int_0^t (A(s) - D(s))\,\mathrm{d}s, \tag{2.27}$$

where we use that $L(t) = A(t) - D(t)$ is the total number of jobs in the system at time t, c.f. Figure 2.5. Observe from the second equation that $\int_0^t L(s)\mathrm{d}s$ is the area enclosed between the graphs of $\{A(t)\}$ and $\{D(t)\}$. Assuming the limit exists for $t \to \infty$, we define

$$\mathbb{E}[L] = \lim_{t\to\infty} \frac{1}{t}\int_0^t L(s)\,\mathrm{d}s \tag{2.28}$$

Observe that, notwithstanding that the symbols are the same, this expectation need not be the same as (2.25), c.f. on of the exercises below. Next, define the following probability as the *time-average fraction of time the system contains at most m jobs*:

$$\mathbb{P}\{L \le m\} = \lim_{t\to\infty} \frac{1}{t}\int_0^t \mathbb{1}_{L(s)\le m}\,\mathrm{d}s. \tag{2.29}$$

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

It is evident how to formulate the counterparts for the waiting time process $\{W(t)\}$.

Exercise 2.8.1 Design a queueing system in which the load as seen the server is very different from what the customers see. Show that for such a queueing system typically

$$\lim_{t\to\infty} \frac{1}{t}\int_0^t L(s)\,\mathrm{d}s \neq \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} L_k.$$

hence, in general:

$$\lim_{t\to\infty} \int_0^t \mathbb{1}_{L(s)\le m}\,\mathrm{d}s \neq \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{L_k\le m},$$

Exercise 2.8.2

1. Assume that $X_k = 10$ minutes and $S_k = 11$ minutes for all k, i.e., X_k and S_k are deterministic and constant. What are λ and μ? Compute A_k, W_k, D_k and L_k. How do these numbers behave as a function of k?.
2. Yet another simple case is to take $X_k = 10$ minutes and $S_k = 9$ minutes for all k. Answer the same questions as in the previous part.

Exercise 2.8.3 Consider a discrete-time model of a queueing system, as we developed in Section 2.4. In that case batches of jobs can arrive in a period, and the average number of customers that *see upon arrival* more than m customers in the system cannot be defined as (2.26). Provide a better definition.

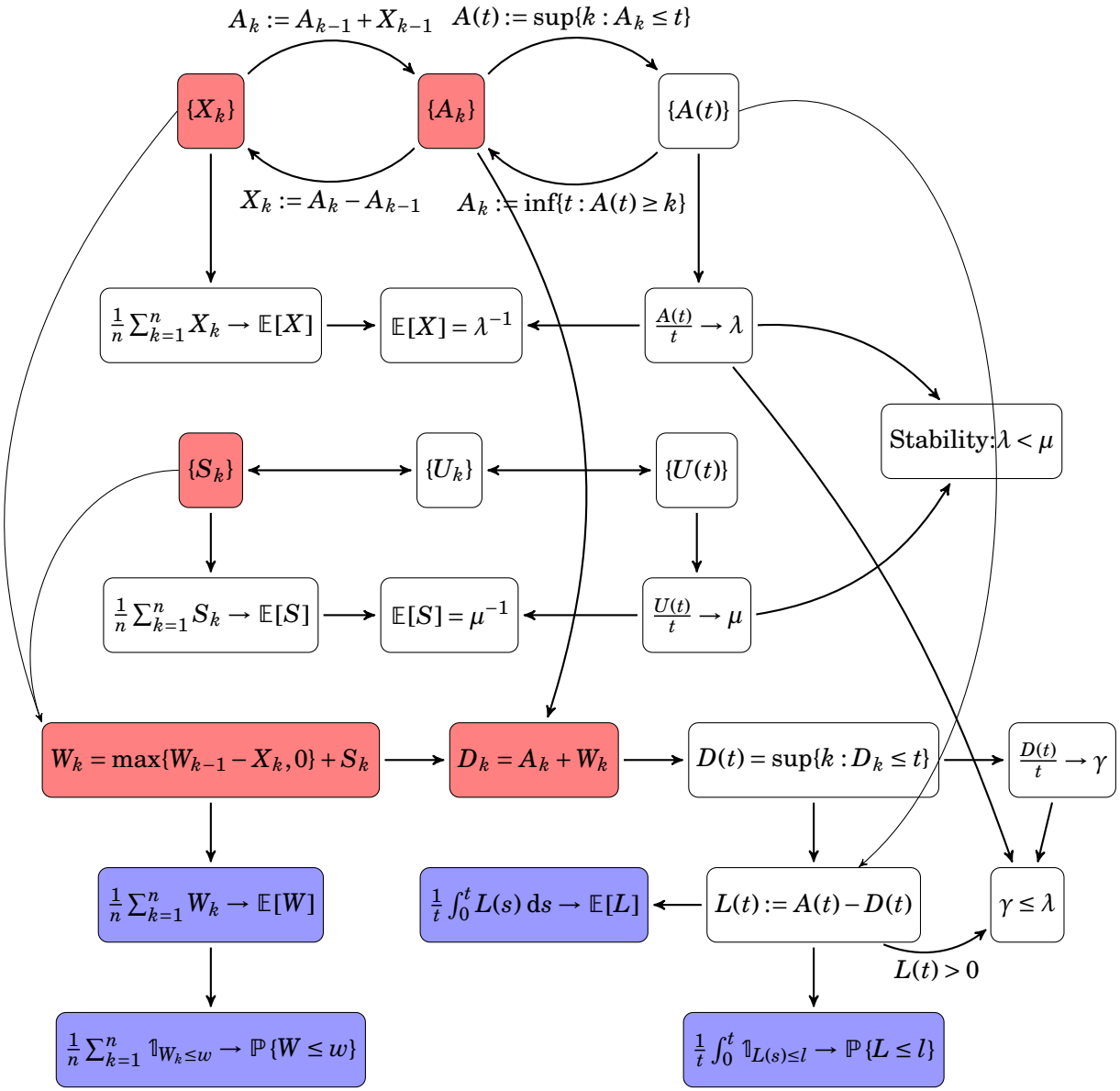


Figure 2.9: Here we sketch the relations between the construction of the $G/G/1$ queue from the primary data, i.e., the interarrival times $\{X_k; k \ge 0\}$ and the service times $\{S_k; k \ge 0\}$, and different performance measure.

2.9 Level-Crossing and Balance Equations

Consider a system at which customers arrive and depart in single entities, such as customers in a shop or jobs at some machine. If the system starts empty, then we know that the number $L(t)$ is the system at time t is equal to $A(t) - D(t)$, see Figure 2.10. Let us denote an arrival as an 'up-crossing' and a departure as a 'down-crossing'. Then, clearly $L(t)$ is the number of up-crossings up to time t minus the number of down-crossings up to time t. If $L(t)$ remains finite, or more generally $\lim_t L(t)/t = 0$, then it must be that

$$\lambda = \lim_t \frac{A(t)}{t} = \lim_t \frac{D(t)+L(t)}{t} = \lim_t \frac{D(t)}{t} + \lim_t \frac{L(t)}{t} = \gamma.$$

Hence, when $L(t)/t \to 0$, the *up crossing rate* $\lim_t A(t)/t = \lambda$ is equal to the *down-crossing rate* $\lim_t D(t)/t = \gamma$. We will generalize these notions of up- and downcrossing in this section to derive the *stationary*, also know as *long-run time average* or *steady-state*, distribution of the number of people in queue.

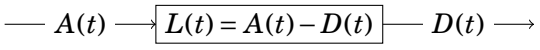


Figure 2.10: What goes in the box, $A(t)$, and has not yet left, $D(t)$, must be in the box, hence $L(t) = A(t) - D(t)$.

Define

$$A(n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t} \mathbb{1}_{L(A_k -)=n}, \tag{2.30a}$$

$$Y(n,t) = \int_0^t \mathbb{1}_{L(s)=n} \, \mathrm{d}s, \tag{2.30b}$$

$$p(n,t) = \frac{1}{t} \int_0^t \mathbb{1}_{L(s)=n} \, \mathrm{d}s = \frac{Y(n,t)}{t}, \tag{2.30c}$$

respectively, so that $A(n,t)$ is the number of arrivals up to time t that saw n customers in the system at their arrival, $Y(n,t)$ is the total time the number in the system $L(s) = n$ during $[0,t]$ so that $p(n,t)$ is the fraction of time that $L(s) = n$ in $[0,t]$. (Recall in the definition of $A(n,t)$ that $L(t)$ is *right-continuous*.)

The above definitions may seem a bit abstract, but they obtain an immediate interpretation when relating them to applications. To see this, we discuss two examples.

Consider the sorting process of post parcels at a distribution center of a post delivery company. Each day tens of thousands of incoming parcels have to be sorted to their final destination. In the first stage of the process, parcels are sorted to a region in the Netherlands. Incoming parcels are deposited on a conveyor belt. From the belt they are carried to outlets (chutes), each chute corresponding to a specific region. Employees take out the parcels from the chutes and put the parcels in containers. The arrival rate of parcels for a certain chute may temporarily exceed the working capacity of the employees, as such the chute serves as a queue. When the chute overflows, parcels are directed to an overflow container and are sorted the next day. The target of the sorting center is to deliver at least a certain percentage of the parcels within one day. Thus, the fraction of parcels rejected at the chute should remain small.

Suppose a chute can contain at most 20 parcels, say. Then, each parcel on the belt that 'sees' 20 parcels in its chute will be blocked. Let $L(t)$ be the number of parcels in the chute

at time t. Then, $A(20,t)$ as defined in Eq. (2.30a) is number of *blocked parcels* up to time t, and $A(20,t)/A(t)$ is the fraction of rejected parcels. In fact, $A(20,t)$ and $A(t)$ are continuously tracked by the sorting center and used to adapt employee capacity to control the fraction of rejected parcels. Thus, in simulations, if one want to estimate loss fractions, $A(n,t)/A(t)$ is the most natural concept to consider.

For the second example, suppose there is a cost associated with keeping jobs in queue. Let w be the cost per job in queue per unit time so that the cost rate is nw when n jobs are in queue. But then is $wnY(n,t)$ the total cost up to time t to have n jobs in queue, hence the total cost up up to time t

$$C(t) = w \sum_{n=0}^{\infty} nY(n,t),$$

and the average cost is

$$\frac{C(t)}{t} = w \sum_{n=0}^{\infty} n \frac{Y(n,t)}{t} = w \sum_{n=0}^{\infty} np(n,t).$$

All in all, the concepts developed above have natural interpretations in practical queueing situations; they are useful in theory and in simulation, as they relate the theoretical concepts to actual measurements.

Continuing with the above notions we define in turn

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

as the *arrival rate in state n*. Similarly, denoting by

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

the number of departures up to time t that leave n customers behind, we have that

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

is *the departure rate from state* $n+1$. (It is easy to get confused here: to leave n jobs behind, the system must contain $n+1$ jobs just prior to the departure.) Figure 2.11 shows how $A(n,t)$ and $\lambda(n)$ relate to $D(n+1,t)$ and $\mu(n)$.

Observe that customers arrive and depart as single units. Thus, if $\{T_k\}$ is the ordered set of arrival and departure times of the customers, then $L(T_k) = L(T_k-) \pm 1$. But then we must also have that $|A(n,t) - D(n,t)| \le 1$ (Think about this.). From this observation it follows immediately that

$$\lim_{t\to\infty} \frac{A(n,t)}{t} = \lim_{t\to\infty} \frac{D(n,t)}{t}. \tag{2.31}$$

With this equation we can obtain two nice and fundamental identities. The first we develop now; the second follows in Section 2.12.

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

$$\frac{A(n,t)}{t} = \frac{A(n,t)}{Y(n,t)} \frac{Y(n,t)}{t} \to \lambda(n)p(n), \tag{2.32a}$$

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

$$\frac{D(n,t)}{t} = \frac{D(n,t)}{Y(n+1,t)} \frac{Y(n+1,t)}{t} \to \mu(n+1)p(n+1). \tag{2.32b}$$

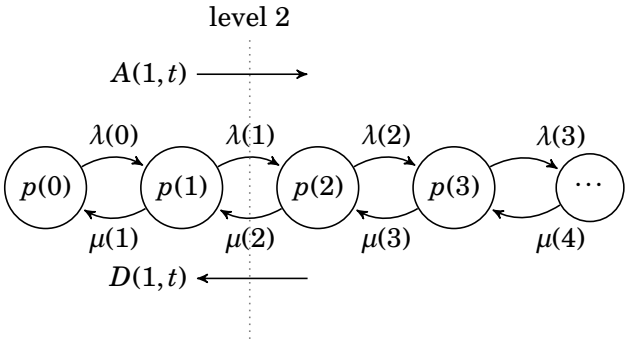


Figure 2.11: $A(1,t)$ counts the number of jobs up to time t that saw 1 job in the system; thus, right after the arrival of such a job the system contains 2 jobs. Similarly, $D(1,t)$ counts the number of departures that leave 1 job behind. Each time $A(1,t)$ increases by one or $D(1,t)$ decreases by one, level 2 (the dotted line separating states 1 and 2) is crossed. The number of times this level is crossed from below must be the same (plus or minus 1) the number of times it is crossed from above.

Combining this with (2.31) we arrive at *the level-crossing equations*

$$\lambda(n)p(n) = \mu(n+1)p(n+1). \tag{2.33}$$

This result turns out to be exceedingly useful. Once we can specify $\lambda(n)$ and $\mu(n)$, we can compute the long-run fraction of time $p(n)$ that the system contains n jobs. To see this, rewrite the above into

$$p(n+1) = \frac{\lambda(n)}{\mu(n+1)}p(n). \tag{2.34}$$

Thus, if we have $p(n)$ we can compute $p(n+1)$, and so on. In other words, if $p(0)$ is known, than $p(1)$ follows, from which $p(2)$ follows, and so on. A straightaway iteration then leads to

$$p(n+1) = \frac{\lambda(n)\lambda(n-1)\cdots\lambda(0)}{\mu(n+1)\mu(n)\cdots\mu(1)}p(0). \tag{2.35}$$

Finally, to determine $p(0)$ we can use the fact that numbers $p(n)$ have to be normalized. Let the *normalization constant* be given by

$$G = 1 + \frac{\lambda(0)}{\mu(1)} + \frac{\lambda(0)\lambda(1)}{\mu(1)\mu(2)} + \cdots. \tag{2.36}$$

Then, $p(0) = G^{-1}$ and $p(n)$ follows from Eq. (2.35).

Let us now express a few important performance measures in terms of $p(n)$: the average number of items $\mathbb{E}[L]$ in the system and the fraction of time $\mathbb{P}\{L \le n\}$ the system contains at least n jobs. As $L(s)$ counts the number of jobs in the system at time s (thus $L(s)$ is an integer),

$$L(s) = \sum_{n=0}^{\infty} n \, \mathbb{1}_{L(s)=n}.$$

With this we can write for the time-average number of jobs in the system

$$\mathscr{L}(t) = \frac{1}{t}\int_0^t \left(\sum_n n \, \mathbb{1}_{L(s)=n}\right) \mathrm{d}s = \sum_n \frac{n}{t}\int_0^t \mathbb{1}_{L(s)=n} \, \mathrm{d}s, \tag{2.37}$$

where we interchange the integral and the summation[5]. It then follows from Eq. (2.30c) that

$$\mathscr{L}(t) = \sum_n n\, p(n,t).$$

Finally, assuming that the limit $p(n,t) \to p(n)$ exists as $t \to \infty$ (and that the summation and limit can be interchanged in the above), it follows that

$$\mathscr{L}(t) \to \sum_{n=0}^{\infty} np(n) = \mathbb{E}[L], \text{ as } t \to \infty.$$

In words, $\mathscr{L}(t)$ converges to the long-run time average $\mathbb{E}[L]$ of the number in the system. In a loose sense we can say that $\mathbb{E}[L]$ is the average number in the system as perceived by the *server*. (Recall that this is not necessarily the same as what *arriving* jobs 'see'). Similarly, the probability that the system contains at least n jobs is

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

From the above we conclude that from the probabilities $p(n)$, more specifically, from a specification of $\lambda(n)$ and $\mu(n)$, we can compute numerous performance measures. Thus, determining $p(n)$ from Eqs. (2.34) and (2.36) for concrete queueing examples is the task of the next sections.

Before we embark on this task, we note that the level-crossing cannot always be used as we do here. The reason is that it is not always natural, or easy, to decompose the state space into two disjoint parts. For a more general approach, we focus on a single state and count how often this state is entered and left, c.f. Figure 2.12. Specifically, define

$$I(n,t) = A(n-1,t) + D(n,t),$$

as the number of times the queueing process enters state n either due to an arrival from state $n-1$ or due to a departure leaving n jobs behind. Similarly,

$$O(n,t) = A(n,t) + D(n-1,t),$$

is counts how often state n is left either by an arrival in state n or a departure to state $n-1$.

Of course, $|I(n,t) - O(n,t)| \leq 1$. Thus, from the fact that

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

and

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

we get that

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

These equations hold for any $n \geq 0$ and are known as the *balance equations*. We will use these equations when studying queueing systems in which level crossing cannot be used, for instance for queueing networks.

Again, just by using properties that hold along any sensible sample path, i.e., counting differences, we obtain very useful statistical/probabilistic results after taking limits.

[5] This is allowed as the integrand is non-negative. More generally, the interested reader should check Fubini's theorem.

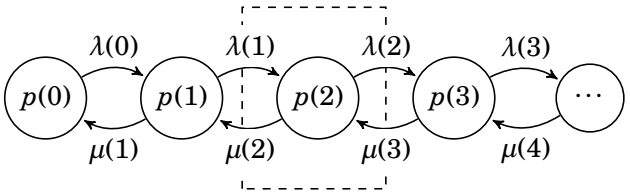


Figure 2.12: For the balance equations count how often a box around a state is crossed from inside and outside. On the long run the entering and leaving rates should be equal. For the example here, the rate out is $p(2)\lambda(2)+p(2)\mu(2)$ while the rate in is $p(1)\lambda(1)+p(3)\mu(3)$.

Exercise 2.9.1 Should we take $L(A_k-)=n$ or $L(A_k)$ in the definition of $A(n,t)$?

Exercise 2.9.2 Should we take $D(n-1,t)$ or $D(n,t)$ in the definition of $\mu(n,t)$?

Exercise 2.9.3 What is the difference between $A(n,t)$ and $A(t)$?

Exercise 2.9.4 Show that $A(n,t)\le A(t)$.

Exercise 2.9.5 Suppose that $\lambda>\gamma$, i.e., the system is not rate-stable. Why is then $\lim_{t\to\infty}\frac{A(n,t)}{t}=0$?

Exercise 2.9.6 Is it always true that $\lambda=\lim_t\frac{A(t)}{t}=\lim_t\frac{A(n,t)}{t}$.

Exercise 2.9.7 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 $Y(n,t)$.What acronym would describe this queueing situation?

2.10 M/M/1 queue

In the $M/M/1$ queue, one server serves jobs arriving with exponentially distributed interarrival times and each requiring an exponentially distributed processing time. With Eq. (2.34), i.e., $\lambda(n)p(n)=\mu(n+1)p(n+1)$ we can derive a number of important results for this queue.

For the $M/M/1$ queue it is reasonable to assume that $\lambda(n)=\lambda$ and $\mu(n)=\mu$, so that from (2.34),

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

as $\rho=\lambda/\mu$. Since this holds for any $n\ge 0$, it follows with recursion that

$$p(n+1)=\rho^{n+1}p(0).$$

Then, from the normalization condition

$$1=\sum_{n=0}^{\infty}p(n)=p(0)\sum_{n=0}^{\infty}\rho^n=\frac{p(0)}{1-\rho},$$

so

$$p(0)=1-\rho,\qquad\qquad p(n)=(1-\rho)\rho^n. \tag{2.38}$$

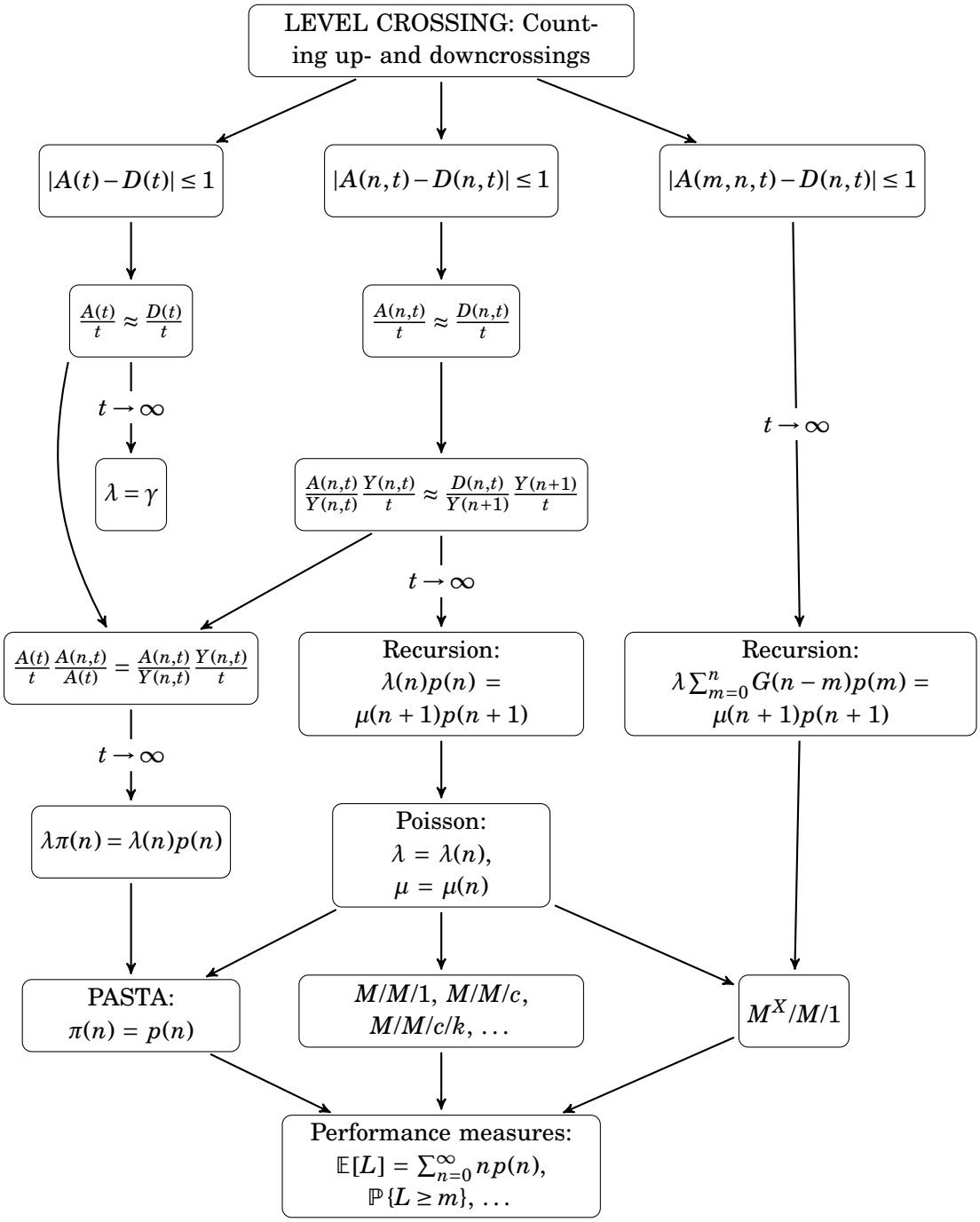


Figure 2.13: With level-crossing arguments we can derive a number of useful relations. This figure presents an overview of these relations that we derive in this and the next sections.

How can we use these equations? First, note that $p(0)$ must be the fraction of time the server is idle. Hence, the fraction of time the server is busy, i.e., the utilization, is

$$1 - p(0) = \rho = \sum_{n=1}^{\infty} p(n).$$

Here the last equation has the interpretation of the fraction of time the system contains at least 1 job. Next, in the exercises below we derive that

$$\mathbb{E}[L] = \frac{\rho}{1-\rho}. \tag{2.39}$$

and

$$\mathbb{P}\{L \geq n\} = \rho^n. \tag{2.40}$$

Let us interpret these expressions. First of all, we only need estimates of λ and μ to characterize the utilization of the server, the average queue length and the queue length distribution. In fact, only the ratio $\rho = \lambda/\mu$ is required. As such, assuming that interarrival times and services are exponentially distribtued, by measuring the fraction of time the server is busy, we obtain an estimate for $1 - p(0)$, hence for ρ. With this, we can estimate $\mathbb{E}[L]$ and $\mathbb{P}\{L \geq n\}$. Next, the fact that $\mathbb{E}[L] \sim (1-\rho)^{-1}$ for $\rho \to 1$ implies that the average waiting time *increases asymptotically fast* to infinity when $\rho \to 1$. In practical terms, this means that striving for a high load results in (very) long average waiting times. As a consequence, this formula tells us that, in the design and operation of queueing systems, we should avoid the situation in which $\rho \approx 1$. Thus, we need to make a trade-off between the server utilization $1 - p(0) = \rho$ and the waiting time. In high load regimes, the server is used efficiently, but the queue lengths are excessive. In any sensible system, we should keep ρ quite a bit below 1, and accept lower utilization then might seem necessary.

The expression for $\mathbb{P}\{L \geq n\}$ shows that the probability that the queue length exceeds some threshold decreases exponentially fast (for any reasonable ρ). Moreover, if we make the simple assumption that customers decide to leave (or rather, not join) the system when the queue is longer than 9 say, then $\mathbb{P}\{L \geq 10\} = \rho^{10}$ is an estimator of the fraction of customers lost. Again, for the $M/M/1$ queue, the knowledge of ρ suffices to compute this estimate.

In the context of inventory theory these equations are particularly useful, see Question 3.

Exercise 2.10.1 Derive expression 2.39 for the average number of jobs in an $M/M/1$ queue.

Exercise 2.10.2 Derive expression 2.40.

Exercise 2.10.3 Customers of, for example, a fast-food restaurant or a production facility, prefer to be served from stock. For this reason such companies often use a 'produce-up-to' policy: When the on-hand inventory I is equal or lower than some threshold r, the company produces items until the inventory level equals $r+1$ again. The level r is the known as the reorder level.

Suppose that customers arrive as a Poisson process with rate λ and the production times of single items are i.i.d. and exponentially distributed with parameter μ. Assume also that customers that cannot be served from on-hand stock are backlogged, that is, they wait until their item has been produced. What are the average on-hand inventory level, the average number of customer in backlog, and the fraction of customers that are backlogged?

If h is the cost per item per unit time in stock and b the cost per customer per unit time in backlog and π the cost per customer backlogged, what are the average costs of using a reorder level r?

What is the optimal reorder level r^*, i.e., the level that achieves minimal average cost? 

2.11 M(n)/M(n)/1 Queue

As it turns out, many more single-server queueing situations can be modeled and analyzed by making a judicious choice of $\lambda(n)$ and $\mu(n)$ in (2.34), to wit, $\lambda(n)p(n) = \mu(n+1)p(n+1)$. For these queueing systems we just present the results. In the exercises we ask you to derive the formulas.

For the $M/M/1/K$, i.e., a system that cannot more than K jobs, take

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n \le K, \\ 0, & \text{if } n > K, \end{cases}$$
$$\mu(n) = \mu.$$

Then,

$$p(n) = \frac{\rho^n}{G} \tag{2.41a}$$
$$G = \frac{1-\rho^{K+1}}{1-\rho}, \tag{2.41b}$$
$$P_{\text{loss}} = \mathbb{P}\{L = K\} = \frac{\rho^K}{G}. \tag{2.41c}$$

For the $M/M/c$ queue we can take

$$\lambda(n) = \lambda,$$
$$\mu(n) = \begin{cases} n\mu, & \text{if } n < c, \\ c\mu, & \text{if } n \ge c. \end{cases}$$

This model is also known as the *Erlang C*-formula. Define the load as 

$$\rho = \frac{\lambda}{c\mu}.$$

Then,

$$p(n) = \frac{1}{G}\frac{(c\rho)^n}{n!}, n = 0, \ldots, c-1 \tag{2.42a}$$
$$p(n) = \frac{1}{G}\frac{c^c\rho^n}{c!}, n = c, c+1, \ldots \tag{2.42b}$$
$$G = \sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{(1-\rho)c!}, \tag{2.42c}$$
$$\mathbb{E}[L_Q] = \sum_{n=c+1}^{\infty}(n-c)p(n) = \frac{(c\rho)^c}{c!G}\frac{\rho}{(1-\rho)^2}, \tag{2.42d}$$
$$\mathbb{E}[L_S] = \sum_{n=01}^{c-1} np(n) = \frac{\lambda}{\mu}. \tag{2.42e}$$

By taking the limit $c \to \infty$ we obtain the $M/M/\infty$ queue, in which case

$$\rho = \frac{\lambda}{\mu} \tag{2.43a}$$

$$p(n) = \frac{1}{G}\frac{\rho^n}{n!} \tag{2.43b}$$

$$G = e^{\rho} \tag{2.43c}$$

$$\mathbb{E}[L] = \mathbb{E}[L_S] = \rho. \tag{2.43d}$$

A modelk, often used to determine the number of beds at hospitals, is the $M/M/K/K$ model, also known as the Erlang B-formula. Watch-out, here $M/M/K/K$ means that at most K-jobs fit into the entire system. In our notation, we should write $M/M/K/0$ where the 0 means that the queue length is bounded by zero. As mentioned before, the meaning of the last letter is not defined consistently in the literature.

Finally, we consider queues with *balking*, that is, a queue in which customers leave when they find the queue too long at the moment they arrive. One simple choice to model customer balking could be as follows,

$$\lambda(n) = \begin{cases} \lambda, & \text{if } n = 0, \\ \lambda/2, & \text{if } n = 1, \\ \lambda/4, & \text{if } n = 2, \\ 0, & \text{if } n > 2, \end{cases}$$

and $\mu(n) = \mu$.

In these examples we made a subtle implicit assumption, on which we elaborate in Section 2.12. To make the problem clear, observe in example 3 that balking customers *decide at the moment they arrive*, in other words, based on what they 'see upon arrival' they either join or leave. In yet other words, they make decisions based on the state of the system at arrival moments, not on time-averages. However, the notion of $p(n)$ is a long-run *time-average*, and is typically not the same as what customers 'see upon arrival'. As a consequence, the performance measure $\mathbb{P}\{L \le n\}$ is also not necessarily in accordance with the perception of customers. To relate these two 'views', i.e., time-average versus observer-average, we need a new concept, *PASTA*, to be developed in the next section.

Exercise 2.11.1 Consider the $M/M/2/1$ queue with arrival rate λ and service rate μ.

1. Derive the level-crossing equations for this queueing system.
2. Derive a simple and closed form expressions for the state probabilities in steady state.
3. Show with a figure how to use level crossing arguments to find the steady state probabilities.

Exercise 2.11.2 Derive the steady state probabilities $p(0), p(1), \ldots$, for the $M/M/k/K$ queue. You do not have to compute the normalization constant G.

Exercise 2.11.3

1. Derive Eq. (2.41)
2. Show that as $K \to \infty$, the performance measures of the $M/M/1/K$ converge to those of the $M/M/1$ queue.

Exercise 2.11.4 Derive Eq.(2.42)

Exercise 2.11.5

1. Show that the $M/M/c$ queue converges to the $M/M/\infty$ queue as $c \to \infty$.
2. Show that the $M/M\infty$ queue is stable for any finite λ.
3. Why is $\mathbb{E}[L] = \rho$ for the $M/M/\infty$ queue?

Exercise 2.11.6

1. Derive the steady state probabilities $p(n)$ for a single-server queue with a finite calling population with N members, i.e., jobs that are in service cannot arrive to the system.
2. Check the answer you obtained for the cases $N = 1$ and $N = 2$. Interpret the results.
3. Derive the steady state probabilities $p(n)$ for a queue with a finite calling population with N members and N servers, i.e., the number of servers in the queueing system is equal the size of the calling population.

Exercise 2.11.7 (Hall 5.1) Give two examples of systems that ordinarily have a finite buffer size. Give two examples of systems that have a finite calling population.

Exercise 2.11.8 In what way is a queueing system with balking, at level b say, different from a queueing system with finite calling population of size b?

2.12 Poisson Arrivals See Time Averages

Suppose the following limit exists:

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

thus, $\pi(n)$ is the long-run fraction of jobs that observe n customers in the system at the moment of arrival. It is natural to ask whether $\pi(n)$ and $p(n)$ are related, that is, what is the relation between what customers see upon arrival and the time-average behavior of the system?

We can make some progress with this question by rewriting $\pi(n)$ in the following way. Since $A(t) \to \infty$ as $t \to \infty$, we have that

$$\begin{aligned} \pi(n) &= \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{A(t)} \mathbb{1}_{L(A_k-)=n} \\ &= \lim_{t\to\infty} \frac{1}{A(t)} \sum_{k=1}^{\infty} \mathbb{1}_{A_k\le t, L(A_k-)=n} \\ &= \lim_{t\to\infty} \frac{A(n,t)}{A(t)}, \end{aligned} \tag{2.44}$$

where we use (2.30a) in the last row. Considering that

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

it follows from this, Eq. (2.18).

$$\lambda\pi(n) = \lim_{t\to\infty}\frac{A(t)}{t}\lim_{t\to\infty}\frac{A(n,t)}{A(t)} = \lim_{t\to\infty}\frac{A(t)}{t}\frac{A(n,t)}{A(t)}. \tag{2.45}$$

Since we have, by Eq. (2.32), that

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

it follows that

$$\lambda\pi(n) = \lambda(n)p(n). \tag{2.46}$$

So, why is this useful? In fact, we obtain from this the important insight that when $\lambda(n) = \lambda$ for all n, the averages $\pi(n)$ as *observed by customers are the same as the averages observed by the server*, i.e., the time-average $p(n)$. In other words, only when the arrival rate does not depend on the state of the system, we have that $\lambda(n) = \lambda$ for all n. Of course, this property is not satisfied for any general queueing system, c.f. Exercise 2, however, it is true when the arrival process is Poisson. This fact is typically called *PASTA*: Poisson Arrivals See Time Averages. Thus, when customers arrive in accordance to a Poisson process (so that the inter-arrival times are exponentially distributed), it must be that $\pi(n) = p(n)$, hence, from (2.38), we see that

$$\pi(n) = \rho^n\pi(0) = (1-\rho)\rho^n.$$

There is a further relation between $\pi(n)$ and the statistics of the system as obtained by the departures, i.e., $\delta(n)$, to be defined presently. For this we turn again to Eq. (2.31), i.e., $|A(n,t) - D(n,t)| \le 1$. To obtain Eq. (2.33) we divided both sides of this equation by the time the system spends in a certain state. We can also use another form:

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

Taking limits at the left and right, we see again that the left hand becomes $\lambda\pi(n)$. For the right hand side, we use Eq. (2.20) and define, analogous to (2.44),

$$\delta(n) = \lim_{t\to\infty}\frac{D(n,t)}{D(t)}. \tag{2.47}$$

Thus, $\delta(n)$ is the long-run fraction of jobs that leave n jobs *behind*. Clearly, then, if the limits exist, the right hand side tends to $\gamma\delta(n)$ as $t \to \infty$. Hence, for (queueing) systems in which customers arrive and leave as single units,

$$\lambda\pi(n) = \gamma\delta(n). \tag{2.48}$$

Moreover, if the system is rate-stable, i.e., the output rate γ is equal to the input rate λ, we obtain

$$\pi(n) = \delta(n). \tag{2.49}$$

This means that the system as seen by arrivals, i.e.,. $\pi(n)$, is the same as what jobs leave behind, i.e., $\delta(n)$.

There is a subtle problem in (2.44) and the derivation of (2.45): $\pi(n)$ is defined as a limit over arrival epochs while in $A(n,t)/t$ we take the limit over time. Now the observant reader might ask why these limits should relate at all. The resolution lies in the *renewal reward theorem*. As this theorem is intuitive and very useful in its own right we state this theorem first, and then apply to show (2.45). For the (simple) proof we refer to El-Taha and Stidham Jr. [1998].

Theorem 2.12.1 (Renewal Reward Theorem, $Y = \lambda X$)**.** Suppose the counting process $\{N(t), t \geq 0\}$ is such that $N(t)/t \to \lambda$ as $t \to \infty$, where $0 < \lambda < \infty$. Let $\{Y(t), t \geq 0\}$ be a non-decreasing right-continuous (deterministic) process. Define $X_k = Y(T_k) - Y(T_{k-1}$, $k \geq 1$, where T_k are the epochs at which N increases, i.e., $N(t) = \{k : T_k \leq t\}$. Then $\lim_t Y(t)/t = Y$ iff $\lim_n n^{-1} \sum_k^n X_k = X$, in which case $Y = \lambda X$.

If we take

$$
\begin{aligned}
Y &= \lim_{t\to\infty} \frac{A(n,t)}{t} = \lim_{t\to\infty} \frac{1}{t} \sum_{k=1}^{\infty} \mathbb{1}_{L(A_k-)=n} \mathbb{1}_{A_k \leq t} \\
X_k &= A(n, A_k) - A(n, A_{k-1}) \\
X = \pi(n) &= \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} X_k(n) = \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}_{L(A_k-)=n} \\
\lambda &= \lim_{t\to\infty} \frac{A(t)}{t}.
\end{aligned}
$$

Eq. (2.45) follows right away from the renewal reward theorem.

In many (stochastic) system the relation $Y = \lambda X$ is useful. In particular, in the optimization of queueing, inventory system, and Markov chains, this relation of often used.

Exercise 2.12.1 Check that the conditions of the renewal reward theorem are satisfied in the above proof of (2.45).

Exercise 2.12.2 To see that $p(n)$ and $\pi(n)$ can be very different, consider a queueing system in which the inter-arrival times $\{X_i\}$ are all identical to 1 hour, and the service times are all identical to $S_i = 59$ minutes.

1. What are ρ, $p(0)$, $p(n)$ for $n \geq 2$. What is $\pi(n)$? What is the time average queue length $\mathbb{E}[L]$? What is the queue length as observed by customers?
2. Check Eq. (2.46) for this case.
3. Consider another (theoretical) queueing system in which each customer requires precisely 1 minute. At the start of each hour, 59 customers arrive. What is $p(n)$, what is $\pi(n)$?

Exercise 2.12.3 Is in general $\pi(n) \geq \delta(n)$?

Exercise 2.12.4 Why is $\mu(n) = \mu$ for the $M/M/1$ queue?

Exercise 2.12.5 Show that

$$\lambda\pi(n) = \lambda(n)p(n) = \mu(n+1)p(n+1) = \mu\delta(n).$$

What is the important condition for this to be true?

2.13 Little's Law

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

$$\mathbb{E}[L] = \lambda \mathbb{E}[W]. \tag{2.50}$$

Here we provide a sketch of the proof, c.f., El-Taha and Stidham Jr. [1998] for the details. In the forthcoming sections we will apply Little's law frequently. Part of the usefulness of Little's law is that it applies to all input-output systems, whether it is a queueing system or an inventory system or some much more general system.

We start with defining a few intuitively useful concepts. Clearly, from (2.27),

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

is the time-average of the number of jobs in the system during $[0,t]$. Observe once again from the second equation that $\int_0^t L(s)\,\mathrm{d}s$ is the area enclosed between the graphs of $A(s)$ and $D(s)$.

The waiting time of the kth job is the time between the moment the job arrives and departs, that is

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

We can actually relate W_k to $\mathscr{L}(t)$, see Figure 2.5. Consider a departure time T at which the system is empty. Observe that $A(T) = D(T)$, as at time T all jobs that arrived up to T also have left. Thus, for all jobs $k \le A(T)$ we have that $D_k \le A(T)$, so that we can replace the integration bounds in the above expression for W_k by

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

Moreover, if $s \le T$,

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

Therefore,

$$\begin{aligned}\int_0^T L(s)\,\mathrm{d}s &= \int_0^T \sum_{k=1}^{A(T)} 1\{A_k \le s < D_k\}\,\mathrm{d}s \\ &= \sum_{k=1}^{A(T)} \int_0^T 1\{A_k \le s < D_k\}\,\mathrm{d}s = \sum_{k=1}^{A(T)} W_k.\end{aligned}$$

Thus, in words, the area between the graphs of $A(s)$ and $D(s)$ must be equal to the total waiting time spent by all jobs in the system until T. From this,

$$\mathscr{L}(T) = \frac{1}{T}\int_0^T L(s)\,\mathrm{d}s = \frac{A(T)}{T}\frac{1}{A(T)}\sum_{k=1}^{A(T)} W_k.$$

Finally, assuming there are an infinite number of times $0 \le T_i < T_{i+1} < \cdots$ at which $A(T_i) = D(T_i)$ and the following limits exist

$$\begin{aligned} T_i &\to \infty \\ \mathscr{L}(T_i) &\to \mathbb{E}[L], \\ \frac{A(T_i)}{T_i} &\to \lambda, \\ \frac{1}{A(T_i)} \sum_{k=1}^{A(T_i)} W_k &\to \mathbb{E}[W], \end{aligned}$$

when $i \to \infty$, we obtain *Little's law*

$$\mathbb{E}[L] = \lambda \mathbb{E}[W].$$

As a useful first application, consider a server of the $G/G/1$ queue as a system by itself. Clearly, jobs depart from the server at rate δ. By rate stability, the arrival rate at the server is therefore also λ. Next, the time a job remains in at the server is $\mathbb{E}[S]$. Thus, the averate number of jobs at the server is $\mathbb{E}[L_S] = \lambda \mathbb{E}[S]$. As $\lambda \mathbb{E}[S] = \rho$, we get $\mathbb{E}[L_S] = \rho$.

Remark 2.13.1. Sometimes (often?) students memorize Little's law in the wrong way. Thus, as an easy check, use the dimensions of the concepts: $\mathbb{E}[L]$ is an average *number*, λ is a *rate*, i.e., *numbers per unit time*, and $\mathbb{E}[W]$ is waiting *time*. *Checking the dimensions* in the formula prevents painfull mistakes.

Note also that $\mathbb{E}[L] \neq L(t)$; to clarify, the expected number of items in the system is not necessarily equal to the number of items in the system at some arbitrary time t. Thus, Little's law need not hold at all moments in time; it is a statement about *averages*.

Exercise 2.13.1 For a given single server queueing system the average number of customers in the system is $\mathbb{E}[L] = 10$, customers arrive at rate λ and are served at rate μ.

1. What is the average time customers spend in the system?
2. Suppose at the moment you join the system, the number of customers in the system is 10. What is your expected time in the system?
3. Why are the answers between these two questions different?

Exercise 2.13.2 Provide a graphical interpration of the proof of Little's law.

Exercise 2.13.3 Little's law can be applied to any system in which jobs arrive and depart. In particular, we can apply it to the system consisting of just the server. What result do you get for this case?

Exercise 2.13.4

1. Which assumptions did we use to prove that $\rho = \lambda \mathbb{E}[S]$?
2. Can you make an arrival process such that $A(t)/t$ does not have a limit?

2.14 Some Useful Identities

With the PASTA property and Little's law we can derive a number of useful and simple results for the $M/G/1$ queue. Recall, to use the PASTA, we need to assume that jobs arrive as a Poisson process.

The fraction of time the server is empty is $1-\rho = p(0)$. By PASTA, $\pi(0) = p(0)$, hence the fraction of customers that enter an empty system is also $1-\rho$.

Suppose that at A_k, i.e., the arrival epoch of the kth job, the server is busy. The remaining service time S_r, is the time between A_k and the epoch the job in service departs. If the server is free at A_k, we set $S_{r,k} = 0$. The limit, provided it exists,

$$\mathbb{E}[S_r] = \lim_{n\to\infty} \frac{\sum_{k=1}^{n} S_{r,k}}{n}$$

is called the average *remaining service time* as seen at arrival epochs. Note that $\mathbb{E}[S_r]$ includes a fraction of jobs that find the server idle. If we need the remaining service time for the jobs that see the server busy, use that a fraction ρ of the jobs sees the server occupied upon arrival, while a fraction $1-\rho$ sees a free server. Therefore

$$\mathbb{E}[S_r] = \rho\,\mathbb{E}[S_r \,|\, S_r > 0] + (1-\rho)\cdot \mathbb{E}[S_r \,|\, S_r = 0] = \rho\,\mathbb{E}[S_r \,|\, S_r > 0], \tag{2.51}$$

since, evidently, $\mathbb{E}[S_r \,|\, S_r = 0] = 0$.

Next, consider the waiting time in queue. It is evident that the expected waiting time for an arriving customer is the expected remaining service time plus the expected time in queue. The expected time in queue, in turn, must be equal to the expected number of customers in queue at an arrival epoch times the expected service time per customer (assuming that service times are i.i.d.). If the arrival process is Poisson, it follows from PASTA that the average number of jobs in queue perceived by arriving customers is also the *time-average* number of jobs in queue $\mathbb{E}\left[L_Q\right]$. Thus, the time an arrival expects to spend in queue is

$$\mathbb{E}\left[W_Q\right] = \mathbb{E}[S_r] + \mathbb{E}\left[L_Q\right]\mathbb{E}[S]. \tag{2.52}$$

Now,, from Little's law we know that $\mathbb{E}\left[L_Q\right] = \lambda\,\mathbb{E}\left[W_Q\right]$. Using this

$$\mathbb{E}\left[W_Q\right] = \mathbb{E}[S_r] + \lambda\,\mathbb{E}\left[W_Q\right]\mathbb{E}[S] = \mathbb{E}[S_r] + \rho\,\mathbb{E}\left[W_Q\right].$$

But this gives for the $M/G/1$ queue that

$$\mathbb{E}\left[W_Q\right] = \frac{\mathbb{E}[S_r]}{1-\rho} = \frac{\rho}{1-\rho}\,\mathbb{E}[S_r \,|\, S_r > 0], \tag{2.53}$$

where we use (2.51) in the last equation. The average waiting time $\mathbb{E}[W]$ in the entire system, i.e., in queue plus in service, is therefore

$$\mathbb{E}[W] = \mathbb{E}\left[W_Q\right] + \mathbb{E}[S] = \frac{\mathbb{E}[S_r]}{1-\rho} + \mathbb{E}[S]..$$

The situation can be significantly simplified for the $M/M/1$ queue as then the service times are also exponential, hence memoryless, implying that $\mathbb{E}[S_r \,|\, S_r > 0] = \mathbb{E}[S]$. Thus, for the $M/M/1$ queue,

$$\mathbb{E}[W] = \mathbb{E}\left[W_Q\right] + \mathbb{E}[S] = \frac{\mathbb{E}[S]}{1-\rho}.$$

Another way to derive the above result is to conclude from PASTA that the expected number of jobs in the system at an arrival is $\mathbb{E}[L]$. Since all these jobs require an expected service time $\mathbb{E}[S]$, including the job in service by the memory-less property, the time in queue is $\mathbb{E}[L]\,\mathbb{E}[S]$. The time in the system is then the waiting time plus the service time, hence,

$$\mathbb{E}[W] = \mathbb{E}[L]\,\mathbb{E}[S] + \mathbb{E}[S] = \lambda\,\mathbb{E}[W]\,\mathbb{E}[S] + \mathbb{E}[S] = \rho\,\mathbb{E}[W] + \mathbb{E}[S], \tag{2.54}$$

where we use Little's law $\mathbb{E}[L] = \lambda\,\mathbb{E}[W]$. Hence,

$$\mathbb{E}[W] = \frac{\mathbb{E}[S]}{1-\rho}.$$

Also,

$$\mathbb{E}\left[W_Q\right] = \mathbb{E}[L]\,\mathbb{E}[S] = \lambda\,\mathbb{E}[W]\,\mathbb{E}[S] = \rho\,\mathbb{E}[W] = \frac{\rho}{1-\rho}\,\mathbb{E}[S] = \frac{\rho^2}{1-\rho}\,\frac{1}{\lambda},$$

which is consistent with our earlier result.

For the average queue length we use Little's law. Then

$$\mathbb{E}[L] = \lambda\,\mathbb{E}[W] = \frac{\lambda\,\mathbb{E}[S]}{1-\rho} = \frac{\rho}{1-\rho},$$
$$\mathbb{E}\left[L_Q\right] = \lambda\,\mathbb{E}\left[W_Q\right] = \frac{\rho^2}{1-\rho}.$$

Finally, the expected number of jobs in service $\mathbb{E}[L_s]$, which is equal to the expected number of busy servers, must be

$$\mathbb{E}[L_s] = \mathbb{E}[L] - \mathbb{E}\left[L_Q\right] = \frac{\rho}{1-\rho} - \frac{\rho^2}{1-\rho} = \rho,$$

again in accordance with our earlier result.

Remark 2.14.1. It is an easy mistake to think that $\mathbb{E}[S_r] = \mathbb{E}[S]$ when service times are exponential. However, as $\mathbb{E}[S_r \,|\, S_r > 0] = \mathbb{E}[S]$ for the $M/M/1$ queue, and $\mathbb{E}[S_r] = \rho\,\mathbb{E}[S_r \,|\, S_r > 0]$ for the $M/G/1$ queue, it follows that

$$\mathbb{E}[S_r] = \frac{\mathbb{E}[S]}{\rho}.$$

Exercise 2.14.1 Try to derive relation (2.51) with sample path arguments.

Exercise 2.14.2

1. Why is this true for the $M/M/1$ queue

$$\mathbb{E}\left[L_Q\right] = \sum_{n=1}^{\infty} (n-1)\pi(n)?$$

2. Derive an expression for $\mathbb{E}\left[L_Q\right]$ based on this observation.

Exercise 2.14.3 Why is Eq. (2.54) not true in general for the $M/G/1$ queue?

Exercise 2.14.4

1. Use the PASTA property to see that the expected waiting time in the system must also be equal to
$$\mathbb{E}[W] = \sum_{n=0}^{\infty} \mathbb{E}\left[W_Q \,|\, N = n\right] \pi(n) + \mathbb{E}[S],$$
where $\mathbb{E}\left[W_Q \,|\, N = n\right]$ is the waiting time in queue given that an arrival sees $N = n$ customers in front of him.
2. Motivate that $\mathbb{E}\left[W_Q \,|\, N = 1\right] = \mathbb{E}[S]$.
3. Combine the above to see that
$$\mathbb{E}[W] = \mathbb{E}[S] \sum_{n=0}^{\infty} n\pi(n) + \mathbb{E}[S] = \mathbb{E}[S]\,\mathbb{E}[L] + \mathbb{E}[S].$$

Exercise 2.14.5 There is distinction between $\mathbb{E}\left[L_Q\right]$, i.e., the time-average queue length, and the expected number of jobs in the system given that the server is busy, i.e., $\mathbb{E}[L \,|\, B]$ is $\mathbb{E}\left[L_Q\right] \neq \mathbb{E}[L \,|\, B]$?

Exercise 2.14.6 Compute the variance of the number of jobs in the system L.

Exercise 2.14.7 Use the PASTA property and the ideas of Section 2.9 to derive for the $M/M/1$ queue that
$$(\lambda + \mu)\pi(n) = \lambda\pi(n-1) + \mu\pi(n+1)$$

Exercise 2.14.8 What would you guess for $\mathbb{E}[S_r \,|\, S_r > 0]$ for the $M/D/1$ queue?

Exercise 2.14.9 (Hall 5.2) After observing a single server queue for several days, the following steady-state probabilities have been determined: $p(0) = 0.4$, $p(1) = 0.3$, $p(2) = 0.2$, $p(3) = 0.05$ and $p(4) = 0.05$. The arrival rate is 10 customers per hour.

1. Determine $\mathbb{E}[L]$ and $\mathbb{E}\left[L_Q\right]$.
2. Using Little's formula, determine $\mathbb{E}[W]$ and $\mathbb{E}\left[W_Q\right]$.
3. Determine $\mathbb{V}[L]$ and $\mathbb{V}\left[L_Q\right]$.
4. Determine the service time and the utilization.

Exercise 2.14.10 (Hall 5.5) An $M/M/1$ queue has an arrival rate of 100 per hour and a service rate of 140 per hour.

1. What is $p(n)$?
2. What are $\mathbb{E}\left[L_Q\right]$, $\mathbb{E}[L]$

Exercise 2.14.11 (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.

1. What are the service rate and utilization?
2. Calculate $\mathbb{E}\left[L_Q\right]$, $\mathbb{E}[L]$ and $\mathbb{E}[W]$.
3. The queue operator would like to provide chairs for waiting customers. He would like to have a sufficient number so that all customers can sit down at least 90 percent of the time. How many chairs should he provide?

Exercise 2.14.12 (Hall 5.7). A single server queueing system is known to have Poisson arrivals and exponential service times. However, the arrival rate and service time are state dependent. As the queue becomes longer, servers work faster, and the arrival rate declines, yielding the following functions (all in units of number per hour): $\lambda(0) = 5$, $\lambda(1) = 3$, $\lambda(2) = 2$, $\lambda(n) = 0, n \geq 3$, $\mu(0) = 0$, $\mu(1) = 2$, $\mu(2) = 3$, $\mu(n) = 4, n \geq 3$. Calculate the state probabilities, i.e., $p(n)$ for $n = 0, \ldots$.

Exercise 2.14.13 (Hall 5.14) An airline phone reservation line has one server and a buffer for two customers. The arrival rate 6 customers per hour, and a service rate of just 5 customers per hour. Arrivals are Poisson and service times are exponential.

1. Estimate $\mathbb{E}\left[L_Q\right]$ and the average number of customers served per hour.
2. Estimate $\mathbb{E}\left[L_Q\right]$ for a buffer of size 5. What is the impact of the increased buffer size on the number of customers served per hour?

Exercise 2.14.14 Consider an $M/M/1$ queue with $\mu = 1.2$ per hour. Make plots of $\mathbb{E}\left[W_Q\right]$ for various values of λ to show the dependency on the arrival rate. The idea of making such plots is to analyze whether the service capacity suffices for a situation in which one doesn't know the arrival rate, but one can control the service rate, e.g., by hiring personel.

Exercise 2.14.15 (Hall 5.3) After observing a queue with two servers for several days, the following steady-state probabilities have been determined: $p(0) = 0.4$, $p(1) = 0.3$, $p(2) = 0.2$, $p(3) = 0.05$ and $p(4) = 0.05$. The arrival rate is 10 customers per hour.

1. Determine $\mathbb{E}[L]$ and $\mathbb{E}\left[L_Q\right]$.
2. Using Little's formula, determine $\mathbb{E}[W]$ and $\mathbb{E}\left[W_Q\right]$.
3. Determine $\mathbb{V}[L]$ and $\mathbb{V}\left[L_Q\right]$.
4. Determine the service time and the utilization.

Exercise 2.14.16 (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.

1. Calculate the steady state probabilities.
2. What are the averate arrival and service rates?
3. Determine $\mathbb{E}[L]$, $\mathbb{E}[L]_Q$, $\mathbb{E}[W]$ and $\mathbb{E}[W]_Q$.

Exercise 2.14.17 (Hall 5.10) A repair/maintenance facility would like to determine how many employees should be working in its tool crib. The service time is exponential, with mean 4 minutes, and customers arrive by a Poisson process with rate 28 per hour. The customers are actually maintenance workers at the facility, and are compensated at the same rate as the tool crib employees.

1. What is $\mathbb{E}[W]$ for $c = 1,2,3$, or 4 servers?
2. How many employees should work in the tool crib?

Exercise 2.14.18 An $M/M/2$ queueing system is very heavily utilized, with an arrival rate of 11 customers per hour, and a service rate of 6 customers per hour per server.

1. Determine $\mathbb{E}\left[L_Q\right]$ and $\mathbb{E}\left[W_Q\right]$.
2. Compare your solution to an $M/M/1$ queue with $\rho = 11/12$. Why are your answers similar or different?

Exercise 2.14.19 Generalize the single-server relation $\mathbb{E}\left[W_Q\right] = \rho\mathbb{E}[S_r \mid S_r > 0] + \mathbb{E}\left[L_Q\right]\mathbb{E}[S]$ to the $M/M/c$ multi-server queue.

We will use this result later to derive algorithms to compute the performance measures for the analysis of closed-queueing network.

2.15 $M^X/M/1$ Queue: Expected Waiting Time

It is not always the case that jobs arrive in single units, they can also arrive as batches. For instance, cars and busses arrive at a fast food restaurant so that the batch size consists of the number of people in the vehicle. In this section we derive a formula to compute the expected queue length for this queueing process.

Assume that jobs arrive as a Poisson process with rate λ and each *job* contains multiple *items*. Let A_k be the arrival time of job k and $A(t)$ the number of job arrivals up to time t. Denote by B_k the number of items that job k brings into the system. We assume that $\{B_i\}$ is an i.i.d. sequence of discrete random variables distributed as a generic random variable B such that $\mathbb{P}\{B = k\} = f(k)$, where $f(k)$ is a given set of probabilities. We write

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

for the *survivor function* of B. We also assume that the service time of each item is exponentially distributed with average $1/\mu$. This queueing situation denoted by the symbol $M^X/M/1$.

The first criterion we must check for the $M^X/M/1$ queue is the stability: the service rate must be larger than the arrival rate of work. To determine the latter, the total number of items $N(t)$

arrived up to time t must be equal to the number of arrivals $A(t)$ up to time t times the batch size of each arrival, i.e.,

$$N(t) = \sum_{k=0}^{\infty} B_k \mathbb{1}_{A_k \leq t} = \sum_{i=k}^{A(t)} B_k.$$

The (stochastic) process $\{N(t)\}$ is known as the *compound Poisson process*. Clearly, the rate at which items arrive is, approximately,

$$\frac{N(t)}{t} = \frac{A(t)}{t} \frac{1}{A(t)} \sum_{i=k}^{A(t)} B_k.$$

As in the limit $A(t)/t \to \lambda$ and $(A(t))^{-1} \sum_{i=k}^{A(t)} B_k \to \mathbb{E}[B]$, we see that

$$\frac{N(t)}{t} \to \lambda \mathbb{E}[B].$$

Thus, for stability, it is necessary that the service rate $\mu = 1/\mathbb{E}[S]$ for items is larger than the rate at which items arrive. Hence we require that the load is such that

$$\rho = \lambda \mathbb{E}[B] \mathbb{E}[S] = \frac{\lambda \mathbb{E}[B]}{\mu} < 1.$$

For the average waiting time, we can use the derivation of $\mathbb{E}\left[W_Q\right] = \mathbb{E}[S_r]/(1-\rho)$ of Section 2.14 for inspiration. Suppose a batch finds $\mathbb{E}[L]$ items in the system upon arrival. Then the expected time the job spends in queue is

$$\mathbb{E}\left[W_{Q,b}\right] = \mathbb{E}[L] \mathbb{E}[S];$$

note that this is not the same as $\mathbb{E}\left[W_Q\right]$, which is the expected time an *item* spends in queue. If B_r is the number of items of the batch currently in service ($B_r = 0$ is the server is idle), and $L_{Q,b}$ the number of batches in queue, then

$$\mathbb{E}[L] = \mathbb{E}\left[L_{Q,b}\right] \mathbb{E}[B] + \mathbb{E}[B_r].$$

With Little's law, $\mathbb{E}\left[L_{Q,b}\right] = \lambda \mathbb{E}\left[W_{Q,b}\right]$, hence

$$\mathbb{E}\left[W_{Q,b}\right] = \lambda \mathbb{E}[S] \mathbb{E}[B] \mathbb{E}\left[L_{Q,b}\right] + \mathbb{E}[S] \mathbb{E}[B_r],$$

hence,

$$\mathbb{E}\left[W_{Q,b}\right] = \frac{\mathbb{E}[S]}{1-\rho} \mathbb{E}[B_r].$$

And then, from the first equation,

$$\mathbb{E}[L] = \frac{\mathbb{E}\left[W_{Q,b}\right]}{\mathbb{E}[S]} = \frac{\mathbb{E}[B_r]}{1-\rho}.$$

Below we prove that

$$\mathbb{E}[B_r] = \rho \frac{\mathbb{E}\left[B^2\right]}{2\mathbb{E}[B]} + \frac{\rho}{2}. \tag{2.55}$$

In Exercise 10 below we rewrite this to

$$\mathbb{E}[L] = \frac{\mathbb{E}[B_r]}{1-\rho} = \frac{1+C_s^2}{2} \frac{\rho}{1-\rho} \mathbb{E}[B] + \frac{1}{2} \frac{\rho}{1-\rho}, \tag{2.56}$$

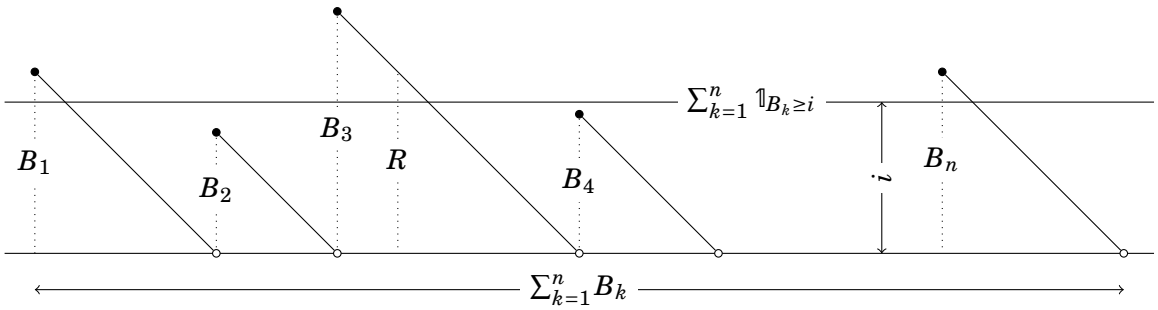


Figure 2.14: The remaining job size as a function of time.

where

$$C_s^2 = \frac{\mathbb{V}[B]}{(\mathbb{E}[B])^2},$$

is the square coefficient of variation of the batch sizes. Observe that this is nearly the same as $M/G/1$ waiting time formula 2.59, a result we derive in Section 2.16.

Thus, to compute the average number of items in the system, we only need to know the first and second moment (or the variance) of the batch size B. Thus, no matter how 'complicated' the distribution of B, when its second moment exists, the average queue length and waiting time can be computed with the above result.

We now turn to proving (2.55) with sample-path arguments and counting; its an elegant line of reasoning. First concentrate on the time the server is busy, i.e., just remove all idle times, and plot the remaining job size during the service of the job, c.f, Figure 2.14. Second, define R as the remaining number of items to be served of the job in service at some arbitrary point in time. Let us show that

$$\mathbb{P}\{R = i\} = \frac{\mathbb{P}\{B \geq i\}}{\mathbb{E}[B]} = \frac{G(i-1)}{\mathbb{E}[B]}.$$

In Figure 2.14 concentrate on level i. Only jobs whose initial batch size is larger or equal to i will produce, during its service, a remaining number of items equal to i. Thus, by counting, we see in Figure 2.14 that $\sum_{k=1}^{n} \mathbb{1}_{B_k \geq i}$ is the number of times there are precisely i remaining items. We also see that $\sum_{k=1}^{n} B_k$ is the total time the server is busy. Thus, the probability that an arbitrary arrival sees i remaining items in service is $\sum_{k=1}^{n} \mathbb{1}_{B_k \geq i} / \sum_{k=1}^{n} B_k$, where we use PASTA to conclude that any item of the $\sum_k B_k$ items is 'chosen' with the same probability. Assuming that the limits exist,

$$\frac{n^{-1} \sum_{k=1}^{n} \mathbb{1}_{B_k \geq i}}{n^{-1} \sum_{k=1}^{n} B_k} \to \frac{G(i-1)}{\mathbb{E}[B]} = \mathbb{P}\{R = i\}, \quad \text{as } n \to \infty.$$

With this expression for $\mathbb{P}\{R = i\}$, the expected remaining time becomes

$$\begin{aligned}
\mathbb{E}[R] &= \sum_{i=1}^{\infty} i \, \mathbb{P}\{R = i\} = \sum_{i=1}^{\infty} i \frac{G(i-1)}{\mathbb{E}[B]} \\
&= \sum_{i=0}^{\infty} (i+1) \frac{G(i)}{\mathbb{E}[B]} = \sum_{i=0}^{\infty} i \frac{G(i)}{\mathbb{E}[B]} + \sum_{i=0}^{\infty} \frac{G(i)}{\mathbb{E}[B]} \\
&= \frac{\mathbb{E}[B^2]}{2\mathbb{E}[B]} - \frac{\mathbb{E}[B]}{2\mathbb{E}[B]} + \frac{\mathbb{E}[B]}{\mathbb{E}[B]} = \frac{\mathbb{E}[B^2]}{2\mathbb{E}[B]} + \frac{1}{2},
\end{aligned}$$

where we use the results of the exercises below.

Finally, recalling that in the above we conditioned on the server being busy, we get that $\mathbb{E}[B_r] = \rho\mathbb{E}[R]$. Combining this with the above expression we obtain (2.55).

Remark 2.15.1. The above derivation for $\mathbb{P}\{R = i\}$ is important for its own sake. It is known as the *remaining lifetime distribution* or the *residual life*. It is worth remembering, in particular the sample path derivation.

Exercise 2.15.1 Why is $\rho = \lambda\mathbb{E}[B]\mathbb{E}[S]$ the appropriate definition of load for the $M^X/M/1$ queue?

Exercise 2.15.2 Relate $f(k)$, the distribution $F(k)$ of the batch size B and the survivor function G. Which of the three alternatives is true: $G(k) = 1 - F(k)$, $G(k) = 1 - F(k-1)$, $G(k) = 1 - F(k+1)$?

Exercise 2.15.3 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 affects the average queueing time. Suppose, for the sake of an example, that the batch size $B = 1, 2$, or 3, with equal probability. Batches arrive at rate 1 per hour. The average processing for an item is 25 minutes. How much does the waiting time decrease if batch sizes are 2 always?

Exercise 2.15.4 In a production enviroment, a machine replenishes an inventory of items (e.g., hamburgers) at a fixed rate of 1 per 3 minutes. If the inventory reaches some level r, the machine stops. Customers arrive at rate of 6 per hour. A customer can buy items in different quanties, $B = 1, 2, 3, 4$, all with equal probability. What is a sensible value for the stopping level r?

Exercise 2.15.5 Use indicator functions to prove that

$$\sum_{n=0}^{\infty} G(n) = \mathbb{E}[B].$$

Exercise 2.15.6 Use indicator functions to prove that

$$\sum_{i=0}^{\infty} iG(i) = \frac{\mathbb{E}[B]^2}{2} - \frac{\mathbb{E}[B]}{2}.$$

Exercise 2.15.7 Use indicator functions to prove that for a continuous random variable S with distribution function F,

$$\mathbb{E}[S] = \int_0^{\infty} x dF = \int_0^{\infty} G(y)\,\mathrm{d}y,$$

where $G(x) = 1 - F(x)$.

Exercise 2.15.8 Use indicator functions to prove that for a continuous random variable S with distribution function F,

$$\mathbb{E}[S]^2 = \int_0^\infty x^2 dF = 2\int_0^\infty yG(y)\,\mathrm{d}y,$$

where $G(x) = 1 - F(x)$.

Exercise 2.15.9 Use that $\mathbb{E}[S] = \int_0^\infty x dF = \int_0^\infty G(y)\,\mathrm{d}y$ to check that $\mathbb{E}[S] = \mu^{-1}$ if $F(x) = 1 - e^{-\mu x}$.

Exercise 2.15.10 Show that

$$\frac{\lambda}{\mu}\mathbb{E}\left[B^2\right] = \rho(1 + C_s^2)\mathbb{E}[B],$$

Exercise 2.15.11 Show that the expression $\mathbb{E}\left[L(M^X/M/1)\right]$ reduces to $\mathbb{E}[L(M/M/1)]$ when the batch size is 1.

Exercise 2.15.12 Compare $\mathbb{E}\left[L(M^X/M/1)\right]$ to $\mathbb{E}[L(M/M/1)]$ when the loads are the same. What do you conclude?

2.16 M/G/1 Queue: Expected Waiting Time

Let's focus on one queue in a supermarket, served by one cashier, and assume that customers do not jockey, i.e., change queue. What can we say about the average waiting time in queue if service times are not exponential, like in the $M/M/1$ queue, but have a more general distribution? One of the celebrated results of queueing theory is the Pollackzek-Khintchine formula by which we can compute the average waiting time formula for the $M/G/1$ queue. In this section we derive this result by means of sample path arguments.

Recall that Eq. (2.53) states that

$$\mathbb{E}\left[W_Q\right] = \frac{\mathbb{E}[S_r]}{1-\rho}$$

This equation is valid for the $M/G/1$ queue, as in the derivation we used the PASTA property but did not make any assumption about the service time distribution, except that it has a finite mean. To use it, we need to compute the average remaining service time for generally distributed service times. This is what we will do here.

Consider the $k$the job of some sample path the $M/G/1$ queueing process. This job requires S_k units of service, let its service time start at time A_k so that it departs the server at time $D_k = A_k + S_k$, c.f. Figure 2.15. Define

$$R_k(s) = (D_k - s)\,\mathbb{1}_{A_k \le s < D_k}$$

as the remaining service time at time s of job k. So see this, observe that when the time $s \in [A_k, D_k)$, the remaining service time until job k departs is $D_k - s$, while if $s \notin [A_k, D_k)$, job k is not in service so it cannot have any remaining service as well.

For the average remaining service time, we can add all remaining service times up to some time t, and then divide by t, specifically,

$$R(t) = \frac{1}{t}\int_0^t \sum_{k=1}^{\infty} R_k(s)\,\mathrm{d}s,$$

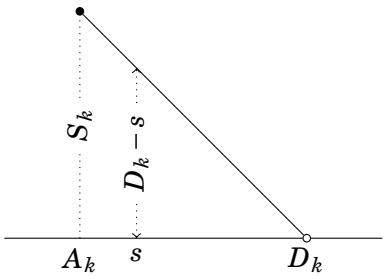


Figure 2.15: Remaining service time.

so that we can define

$$\mathbb{E}[S_r] = \lim_{t\to\infty} R_t,$$

provided this limit exists along the sample path.

Now observe that as t can be somewhere in a service interval, it is easiest to consider $R(t)$ at departure times $\{D_n\}$, as at a time D_n precisely n jobs departed. Thus,

$$R(D_n) = \frac{1}{D_n}\int_0^{D_n}\sum_{k=1}^{\infty} R_k(s)\,\mathrm{d}s = \frac{1}{D_n}\sum_{k=1}^{n}\int_0^{D_n} R_k(s)\,\mathrm{d}s,$$

the interchange being allowed by the positivity of $R_k(s)$. But, if $k < n$ so that $D_k < D_n$,

$$\begin{aligned}\int_0^{D_n} R_k(s)\,\mathrm{d}s &= \int_0^{D_n}(D_k - s)\,\mathbb{1}_{A_k\le s<D_k}\,\mathrm{d}s = \int_{A_k}^{D_k}(D_k - s)\,\mathrm{d}s\\ &= \int_{A_k}^{A_k+S_k}(A_k + S_k - s)\,\mathrm{d}s = \int_0^{S_k}(S_k - s)\,\mathrm{d}s\\ &= \int_0^{S_k} s\,\mathrm{d}s = \frac{S_k^2}{2}.\end{aligned}$$

With this,

$$R(D_n) = \frac{1}{D_n}\sum_{k=1}^{n}\frac{S_k^2}{2} = \frac{n}{D_n}\frac{1}{n}\sum_{k=1}^{n}\frac{S_k^2}{2}.$$

Finally, assuming that along this sample path, $D_n \to \infty$ as $t \to \infty$, $n/D_n \to \gamma$, $\gamma = \lambda$ by rate stability, and the limit $n^{-1}\sum_{k=1}^{n} S_k^2$ exists as $n \to \infty$, we get

$$\mathbb{E}[S_r] = \lim_{D_n\to\infty} R(D_n) = \frac{\lambda}{2}\mathbb{E}\left[S^2\right]. \tag{2.57}$$

In the exercises we ask you to prove that

$$\mathbb{E}[S_r] = \frac{1 + C_s^2}{2}\rho\,\mathbb{E}[S], \tag{2.58}$$

where

$$C_s^2 = \frac{\mathbb{V}[S]}{(\mathbb{E}[S])^2},$$

With the above we have obtained the *Pollackzek-Khintchine* (PK) formula for the average

waiting time in queue:

$$\mathbb{E}\left[W_Q\right] = \frac{1}{1-\rho}\mathbb{E}[S_r] = \frac{1+C_s^2}{2}\frac{\rho}{1-\rho}\mathbb{E}[S]. \tag{2.59}$$

This proves to be a very useful result; a few of the problems below will clarify how to use it.

Exercise 2.16.1 Show from Eq. (2.57) that

$$\mathbb{E}[S_r \,|\, S_r > 0] = \frac{\mathbb{E}\left[S^2\right]}{2\mathbb{E}[S]}$$

Exercise 2.16.2 Show (2.58).

Exercise 2.16.3 Show that $\mathbb{E}[S_r \,|\, S_r > 0] = 1/2\mu$ when S is deterministic and equal to $1/\mu$.

Exercise 2.16.4 Show that $\mathbb{E}[S_r \,|\, S_r > 0] = 2\mu^{-1}/3$ when S is $U[0, 2/\mu]$.

Exercise 2.16.5 Show that $\mathbb{E}[S_r \,|\, S_r > 0] = \mu^{-1}$ when $S \sim \exp(\mu)$.

Exercise 2.16.6 A queueing system receives Poisson arrivals at the rate of 5 per hour. The single server has a uniform service time distribution, with a range of 4 minutes to 6 minutes. Determine $\mathbb{E}\left[L_Q\right]$, $\mathbb{E}[L]$, $\mathbb{E}\left[W_Q\right]$, $\mathbb{E}[W]$.

Exercise 2.16.7 Consider a workstation with just one machine. Jobs arrive, roughly, in accordance to Poisson process, with rate $\lambda = 3$ per day. The average service time $\mathbb{E}[S] = 2$ hours, $C_s^2 = 1/2$, and the shop is open for 8 hours. What is $\mathbb{E}\left[W_Q\right]$?

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

Exercise 2.16.8 (Hall 5.16) The manager of a small firm would like to determine which of two people to hire. One employee is fast, on average, but is somewhat inconsistent. The other is a bit slower, but very consistent. The first has a mean service time of 2 minutes, with a standard deviation of 1 minute. The second has a mean service time of 2.1 minute, with a standard deviation of 0.1 minute. If the arrival rate is Poisson with rate 20 per hour, which employee would minimize $\mathbb{E}\left[L_Q\right]$? Which would minimize $\mathbb{E}[L]$?

Exercise 2.16.9 Show that the Pollackzek-Khintchine formula reduces to $\mathbb{E}\left[W_Q(M/M/1)\right]$ when services are exponential.

Exercise 2.16.10 Compute $\mathbb{E}\left[W_Q\right]$ and $\mathbb{E}[L]$ for the $M/D/1$ queue. Assume that the service time is always T. Compare $\mathbb{E}[L(M/D/1)]$ to $\mathbb{E}[L(M/M/1)]$ where the mean service time is the same in both cases.

Exercise 2.16.11 Compute $\mathbb{E}[L]$ for the $M/G/1$ queue with $G = U[0, A]$, i.e., the uniform distribution on $[0, A]$.

Exercise 2.16.12 Show that for the $M/G/1$ queue, the expected idle time is $\mathbb{E}[I] = 1/\lambda$.

Exercise 2.16.13 Use Exercise 6 to show that for the $M/G/1$ queue, the expected busy time is given by $\mathbb{E}[B] = \mathbb{E}[S]/(1-\rho)$.

Exercise 2.16.14 What is the utilization of the $M/G/1/1$ queue?

Exercise 2.16.15 For the $M/G/1/1$ queue what is the fraction of jobs rejected; what is the fraction of accepted jobs?

Exercise 2.16.16 Why is the fraction of lost jobs at the $M/G/1/1$ queue not necessarily the same as for the $G/G/1/1$ queue? Support your claim with an example.

2.17 $M^X/M/1$ Queue Length Distribution

In this section we extend the analysis of the $M^X/M/1$ batch queue that we started in Section 2.15. We present a numerical, recursive, scheme to compute the stationary distribution of the number of items in queue.

To find the distribution $p(n)$, which is equal to $\pi(n)$ by PASTA, we turn to level-crossing arguments. However, the arguments that lead to the level-crossing equation (2.31) need to generalized. To see this, we consider an example. If $L(t) = 3$, the system contains 3 items (but not necessarily 3 jobs.) Since the server serves single items, down-crossings of level $n = 3$ occur in single units. However, due to the batch arrivals, when a job arrives it typically brings multiple items to the queue. For instance, suppose that $L(A_k-) = 3$, i.e., job k sees 3 items in the system at its arrival epoch. If $B_k = 20$, then $L(A_k) = 3 + 20 = 23$, that is, right after the arrival of job k the system contains 23 items. In this case, at the arrival of job k, all levels between states 3 and 23 are crossed. The left panel in Figure 2.16 demonstrates this in more general terms. Level n can be up-crossed from below from many states, in fact from any level $m < n$. However, it can only be down-crossed from state $n + 1$.

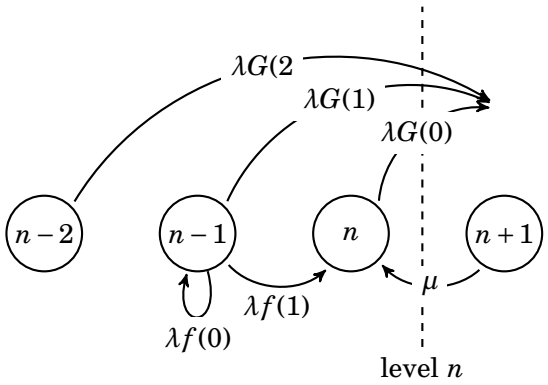


Figure 2.16: Level crossing of level n. Observe that when the system is in state $n-2$, the arrival of any batch larger than 2 ensures that level n is crossed from below. The rate at which such events happen is $\lambda\,\mathbb{P}\{X > 2\} = \lambda G(2)$. Similarly, in state $n-1$ the arrival of any batch larger than one item ensures that level n is crossed, and this occurs with rate $\lambda G(1)$, and so on.

As always with level-crossing arguments, we turn to counting how often level n is upcrossed and downcrossed as a function of time. The downcrossing rate is easy; it is exactly the same as for the $M/M/1$ queue. Thus, from Eq. (2.32b), the downcrossing rate is

$$\frac{D(n,t)}{t} = \frac{D(n,t)}{Y(n+1,t)}\frac{Y(n+1,t)}{t} \to \mu(n+1)p(n+1) = \mu\pi(n+1), \tag{2.60}$$

where we use that $\mu(n+1)=\mu$ by the memoryless property of the service times and PASTA to see that $\pi(n+1)=p(n+1)$. Observe that this part was easy because there is just one way (i.e., there is just one arrow from right to left in Figure 2.16) to downcross level n, namely from $n+1$ to n. Counting upcrossings requires quite some more work.

Let us write $\{L(A_k-)\le n, L(A_k)>n\}$ for the set of sample paths such that the kth job sees n or less items in the system upon arrival (condition $L(A_k-)\le n$) and after the arrival the system contains more than n items (condition $L(A_k)>n$). Thus, $\{L(A_k-)\le n, L(A_k)>n\}$ contains all sample paths whose kth arrival crosses level n from below. We can decompose this event as follows.

$$\begin{aligned}\{L(A_k-)\le n, L(A_k)>n\} &= \{L(A_k-)=n, B_k>0\}\\ &\quad\cup\{L(A_k-)=n-1, B_k>1\}\\ &\quad\cup\dots\\ &\quad\cup\{L(A_k-)=0, B_k>n\}\\ &= \bigcup_{m=0}^{n}\{L(A_k-)=m, B_k>n-m\}.\end{aligned}$$

In other words, paths that upcross level n require that any job that sees $m\le n$ in the system upon arrival must bring a batch larger than $n-m$ items.

Define

$$A(m,n,t)=\sum_{k=1}^{\infty}\mathbb{1}_{A_k\le t}\,\mathbb{1}_{L(A_k-)=m, B_k>n-m}.$$

as the number of jobs that, up to time t, see m in the system upon arrival and have batch size larger than $n-m$. Then the number of times level n is upcrossed up to time t can be written as $\sum_{m=0}^{n}A(m,n,t)$, hence the upcrossing rate up to time t is

$$\frac{1}{t}\sum_{m=0}^{n}A(m,n,t).$$

Simplifying this comes down to finding the right quantities by which to divide. Some experimentation will reveal that the following gives the most relevant insights:

$$\frac{A(m,n,t)}{t}=\frac{A(t)}{t}\frac{A(m,t)}{A(t)}\frac{A(m,n,t)}{A(m,t)}. \tag{2.61}$$

Here $A(t)/t$ and $A(m,t)/A(t)$ have the same meaning as in Section 2.12, hence the first of these fractions converges to λ and the second to $\pi(m)$ as $t\to\infty$. Now, provided the limit exists, we define

$$\lim_{t\to\infty}\frac{A(m,n,t)}{A(m,t)}=\lim_{t\to\infty}\frac{\sum_{k=1}^{\infty}\mathbb{1}_{A_k\le t}\,\mathbb{1}_{L(A_k-)=m, B_k>n-m}}{\sum_{k=1}^{\infty}\mathbb{1}_{A_k\le t}\,\mathbb{1}_{L(A_k-)=m}}=\mathbb{P}\{B>n-m\,|\,L(A_k-)=m\},$$

where $L(A_k-)$ has the interpretation of the number of items in the system at the kth arrival epoch. By *assumption B* and $L(A_k-)$ are independent. Therefore,

$$\lim_{t\to\infty}\frac{A(m,n,t)}{A(n,t)}=\mathbb{P}\{B>n-m\}=G(n-m).$$

By combining the above and making the usual assumptions about the existence of all limits involved we find

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

Thus, Eq. (2.61) has the interpretation that the rate at which level n is crossed from below from state m is equal the rate at which jobs arrive times the fraction of jobs that see m jobs in the system times the fraction of jobs with batch size larger than $n-m$.

The last step is to relate the up- and downcrossing rates. Recall that for the $M/M/1$ the number of upcrossings and downcrossings of level n could not differ more than one, i.e., $|A(n,t) - D(n,t)| \leq 1$. Similarly, for the $M^X/M/1$ queue we have

$$\left| \sum_{m=0}^{n} A(m,n,t) - D(n,t) \right| \leq 1. \tag{2.62}$$

Thus, taking the limit $t \to \infty$ in this equation, we conclude that the upcrossing rate and the downcrossing rate of level n must be the same. Hence,

$$\lambda \sum_{m=0}^{n} G(n-m)\pi(m) = \mu\pi(n+1). \tag{2.63}$$

For the excess probabilities $\mathbb{P}\{L > n\}$ we need to compute the actual probabilities $\pi(n)$ from the recursion (2.63). To apply this, temporarily set $\pi(0) = 1$. Then, for $n = 1$, we have $\mu\pi(1) = \lambda\pi(0) = \lambda$, . Then, since $G(0) = 1$, and writing $\alpha = \lambda/\mu$, the *unnormalized* state probabilities are

$$\begin{aligned}
\pi(1) &= \alpha \\
\pi(2) &= \alpha G(0)\pi(1) + \alpha G(1)\pi(0) = \alpha(\alpha + G(1)) = \alpha^2 + \alpha G(1), \\
\pi(3) &= \alpha[G(0)\pi(2) + G(1)\pi(1) + G(2)\pi(0)] \\
&= \alpha[\alpha^2 + 2\alpha G(1)) + G(2)],
\end{aligned}$$

and so on.

It is left to find the normalization constant. As this recursion does not lead to a closed form expression for $p(n)$, such as Eq. (2.38), we need to use a criterion to stop this iterative procedure. Finding general conditions on when to stop is not directly easy, but a pragmatic approach is to simply stop at some (large) number N and take $G = \sum_{i=0}^{N} \pi(i)$ as the normalization constant. Then $\pi(0) = 1/G$, $\pi(1) = \alpha/G$, and so on.

Remark 2.17.1. An interesting extension is to consider a queueing process with batch services, i.e., the $M/M^Y/1$ queue. Construction a recursion for the steady-state probabilities $p(n)$ for this case is not hard, in fact, mostly analogous to Eq. (2.63). However, solving the recursion is much less simple, hence we will not discuss this further.

Exercise 2.17.1 Show that Eq. (2.63) reduces to $\mu p(n+1) = \lambda p(n)$ for the $M/M/1$ case.

Exercise 2.17.2 Show that $A(n,n,t) = A(n,t)$.

Exercise 2.17.3 Why is Eq. (2.49), i.e., $\pi(n) = \delta(n)$, not true for the $M^X/M/1$ batch queue?

Exercise 2.17.4 Let us use the above recursion Eq. (2.63) for $\pi(n)$ to derive an expression for the expected number of units of work $\mathbb{E}[L]$ in the system. By substituting the recursion Eq. (2.63) in the definition of $\mathbb{E}[L]$ we see that

$$\mu\mathbb{E}[L] = \mu \sum_{n=0}^{\infty} n\pi(n) = \lambda \frac{\mathbb{E}\left[B^2\right]}{2} + \lambda\mathbb{E}[B]\,\mathbb{E}[L] + \lambda \frac{\mathbb{E}[B]}{2}, \tag{2.64}$$

where we use the results of the exercises of Section 2.15 and 4. With this and using the definition for $\rho = \lambda \mathbb{E}[B]/\mu$,

$$(1-\rho)\mathbb{E}[L] = \frac{\lambda}{\mu}\frac{\mathbb{E}\left[B^2\right]}{2} + \frac{\rho}{2}.$$

This is the same result as what we obtained in Section 2.15.

Exercise 2.17.5 (Finite queueing systems) We consider the $M^X/M/1/K$ queue, i.e., a batch queue in which at most K jobs fit into the system. When customers can be blocked, it is necessary to specify an acceptance, or equivalently a rejection, policy. Three common rules are

1. Complete rejection: if a batch does not fit entirely into the system, it will be rejected completely.
2. Partial acceptance: accept whatever fits of a batch, and reject the rest.
3. Complete acceptance: accept all batches that arrive when the system contains K or less jobs, and reject otherwise.

Derive a set of recursions, analogous to Eq. (2.63), to compute $\pi(n)$ for these three different acceptance rules.

Exercise 2.17.6 Implement the recursion (2.63) in a computer program for the case $f(1) = f(2) = f(3) = 1/3$, $\lambda = 1$ and $\mu = 3$.

2.18 M/G/1 Queue Length Distribution

In Section 2.16 we derived the Pollackzek-Khintchine Eq. (2.59) by which we can compute the average waiting time in queue for the $M/G/1$ queue. Then, with Little's law, we therefore also have the average queue length. However, if we need the loss probability $\mathbb{P}\{L > n\}$, we need expressions for the stationary distribution of the number of jobs in the system $p(n) = \mathbb{P}\{L = n\}$. Here we present a set of recursions by which we can compute $p(n)$ for a given arrival rate λ and service distribution F of the service times.

To derive the stationary distribution $p(n)$ we work in steps, and use similar level-crossing arguments as in Section 2.17. As we will see, the argumentation is quite subtle, as it uses an interplay of the PASTA property and the relation of Eq. (2.49) between $\pi(n)$, $p(n)$ and $\delta(n)$. The important idea is to focus on the *departure* moments, as these moments can be considered as restarts of the service times, and on the number of arrivals Y_k during the service time of the kth job. Assume that $f(j) = \mathbb{P}\{Y_k = j\}$ is given and write $G(j) = \mathbb{P}\{Y_k > j\}$.

First we concentrate on a downcrossing of level n, see Figure 2.17. Suppose that the '$k-1$'th job leaves $n+1$ jobs behind after its service completion. Then, if during the service time of the kth job, no other jobs arrive, it must be that job k leaves n jobs behind. Thus, job k realizes a downcrossing of level n only when $L(D_{k-1}) = n+1$ and $Y_k = 0$, i.e.,

$$L(D_{k-1}) = n+1, Y_k = 0 \implies L(D_k) = n$$

Let us next count all downcrossings up to time t. Clearly,

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

is the number of jobs that depart before time t, i.e., $\mathbb{1}_{D_k \le t}$, and whose predecessors leave $n+1$ jobs behind, i.e., $\mathbb{1}_{L(D_{k-1})=n+1}$, and such that no jobs arrive during the service time, i.e., $\mathbb{1}_{Y_k=0}$. . Then, by using the definitions and limits developed in Sections 2.7 and 2.12,

$$\begin{aligned}\lim_{t\to\infty} \frac{D(n+1,0,t)}{t} &= \lim_{t\to\infty} \frac{D(t)}{t}\frac{D(n+1,t)}{D(t)}\frac{D(n+1,0,t)}{D(n+1,t)} \\ &= \gamma\delta(n+1)\lim_{t\to\infty}\frac{D(n+1,0,t)}{D(n+1,t)} \\ &= \gamma\delta(n+1)\mathbb{P}\{Y=0\} \\ &= \gamma\delta(n+1)f(0),\end{aligned}$$

where the last limit follows from the independence of Y_k and $L(D_{k-1})$.

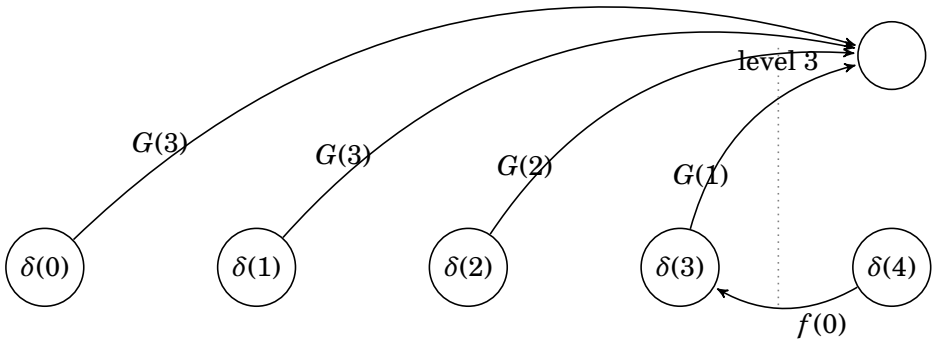


Figure 2.17: Level 3 is crossed from below with rates $\delta(0)G(3)+\delta(1)G(3)+\cdots\delta(3)G(1)$ and crossed from above with rate $\delta(4)f(0)$.

For the upcrossings, suppose first that $L(D_{k-1}) = n > 0$. If there are no arrivals during the service of job k, i.e., $Y_k = 0$, then job k must leave $n-1$ jobs behind. Thus, no upcrossing occurs in this case. Next, if $Y_k = 1$, then job k leaves n jobs behind. Thus, still no upcrossing occurs. In fact, to upcross level n when $L(D_{k-1}) = n > 0$, at least two new jobs must arrive during the service of job k. More generally, if $0 < m \le n$,

$$L(D_{k-1}) = m, Y_k > n-m+1 \implies L(D_k) > n,$$

while if $m = 0$ (think about this),

$$L(D_{k-1}) = 0, Y_k > n \implies L(D_k) > n.$$

Similar to our earlier work, by defining proper counting functions, c.f., Exercise 1, and taking suitable limits, we find for the upcrossing rate that

$$\gamma\delta(0)G(n) + \gamma\sum_{m=1}^{n}\delta(m)G(n-m+1). \tag{2.65}$$

Equating the down-crossings and up-crossing and dividing by γ gives

$$f(0)\delta(n+1) = \delta(0)G(n) + \sum_{m=1}^{n}\delta(m)G(n+1-m). \tag{2.66}$$

Finally, noting that $\pi(n) = \delta(n)$ from (2.49) and the fact that the $M/G/1$ queue length process has one-step transitions, we arrive at

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

Again, we obtain a recursion by which we can compute, iteratively, the state probabilities. Setting $\pi(0) = 1$ at first, $\pi(1)$ follows from the above recursion. Then, with $\pi(0) = 1$ and $\pi(1)$ we find $\pi(2)$, and so on, up to some limit N. Then, set $G = \sum_{n=0}^{N} \pi(n)$ as the normalization constant. As a practical approach to determine whether N is sufficiently large, compute $p(n)$ also for a couple for values above N. If these values decrease when n becomes larger and they are very small compared to the $p(n)$ with $n < N$, N is most surely sufficiently large. However, getting formal bounds on a proper size of N requires more work than we can do here.

Exercise 2.18.1 Provide the details behind the derivation of Eq. 2.65.

Exercise 2.18.2 Suppose that for the $M/G/1$ queue, $L(D_{k-1}) > 0$. Why is $S_k = D_k - D_{k-1}$? If $L(D_{k-1}) = 0$, the time between D_{k-1} and D_k is *not* equal to S_k. What is then the distribution of $D_k - D_{k-1}$?

Exercise 2.18.3 Explain that if the service time is s, then

$$\mathbb{P}\{Y_k = j \,|\, S = s\} = e^{-\lambda s}\frac{(\lambda s)^j}{j!}. \tag{2.67}$$

Exercise 2.18.4 Explain that

$$\mathbb{P}\{Y_k = j\} = \int_0^\infty e^{-\lambda s}\frac{(\lambda s)^j}{j!}\,\mathrm{d}F(s). \tag{2.68}$$

where F is the distribution of the service times.

Exercise 2.18.5 If S is deterministic and equal to s, show that Eq. (2.68) reduces to Eq. (2.67).

Exercise 2.18.6 If $S \sim \exp(\mu)$, show that

$$f(j) = \mathbb{P}\{Y_k = j\} = \frac{\mu}{\lambda+\mu}\left(\frac{\lambda}{\lambda+\mu}\right)^j. \tag{2.69}$$

Exercise 2.18.7 If $S \sim \exp(\mu)$, show that

$$G(j) = \sum_{k=j+1}^{\infty} f(k) = \left(\frac{\lambda}{\lambda+\mu}\right)^{j+1}. \tag{2.70}$$

Exercise 2.18.8 Use Eq. (2.69) to derive a closed-form expression for the queue length distribution for the $M/M/1$ queue.

Exercise 2.18.9 With the above results we can compute Eq. (2.68) for deterministic service times or exponential service times. When, however, the service distribution F is some general distribution function, we need numerical methods to evaluate Eq. (2.68). Can you design a suitable numerical method for this problem?

2.19 A Relation Between the $M^X/M/1$ and $M/G/1$ Queue

There is an interesting relation between the $M^X/M/1$ and the $M/G/1$ queueing systems. As an example, consider the $M^{10}/M/1$ queue in which each job corresponds to the arrival of two items of work. Intuitevely, it is reasonable that this queueing process is quite similar to an $M/G/1$ queue in which each job requires approximately 10 time units. Below we will make this relation precise. We will also use this to find another derivation of the Pollackzek-Khintchine equation, if we consider a certain limiting case.

The basic observation is that by discretizing we can approximate the service time S of the $M/G/1$ queue by the service time of a batch of small units of work. To make this more specific, assume that the distibution of the service time S is given. Use a grid of size $1/n$ and take

$$\mathbb{P}\left\{X^{(n)} = i\right\} = \mathbb{P}\{S \in (i/n, (i+1)/n]\} \tag{2.71}$$

as the distribution of the batch $X^{(n)}$. Let the service time of one unit of the batch be exponentially distributed with mean $1/n$. Thus, if n becomes larger, the service time S is represented by ever larger batches, but such that the service time of each unit becomes ever smaller. These arguments can be made precise in a probabilistic sense, but here we rely on our intuition to conclude that, in some sense, $X^{(n)}/n \xrightarrow{d} S$.

Let us see how this limit can be used to show that the expected waiting in queue $\mathbb{E}[W_Q]$ of the batch queue leads to the Pollackzek-Khintchine equation. Writing $\mathbb{E}[L^{(n)}]$ for the batch queue with grid size $1/n$, we use (2.56) to see that

$$\frac{\mathbb{E}[L^{(n)}]}{n} = \frac{\rho}{1-\rho} \cdot \frac{1}{2}\left(1 + \frac{\mathbb{V}[X^{(n)}]}{(\mathbb{E}[X^{(n)}])^2}\right)\frac{\mathbb{E}[X^{(n)}]}{n} + \frac{1}{2n}\frac{\rho}{1-\rho}. \tag{2.72}$$

By assumption[6]:

$$\frac{X^{(n)}}{n} \to S$$

so that must be that

$$\mathbb{E}\left[\frac{X^{(n)}}{n}\right] \to \mathbb{E}[S]$$

and

$$\mathbb{E}\left[\frac{(X^{(n)})^2}{n^2}\right] \to \mathbb{E}[S^2],$$

as $n \to \infty$. Hence,

$$\begin{aligned}\frac{\mathbb{V}[X^{(n)}]}{n^2} &= \frac{\mathbb{E}[(X^{(n)})^2] - (\mathbb{E}[X^{(n)}])^2}{n^2} = \mathbb{E}\left[\frac{(X^{(n)})^2}{n^2}\right] - \left(\mathbb{E}\left[\frac{X^{(n)}}{n}\right]\right)^2 \\ &\to \mathbb{E}[S^2] - (\mathbb{E}[S])^2 = \mathbb{V}[S].\end{aligned}$$

[6] Here we interchange limits and expectations. A formal approach should pay due attention to the validity of all these interchanges.

from which

$$\frac{\mathbb{V}\left[X^{(n)}\right]}{(\mathbb{E}\left[X^{(n)}\right])^2} = \frac{\mathbb{V}\left[X^{(n)}\right]}{n^2}\frac{n^2}{(\mathbb{E}\left[X^{(n)}\right])^2} \to \frac{\mathbb{V}[S]}{(\mathbb{E}[S])^2} = C_s^2,$$

as $n \to \infty$. Moreover, we expect that

$$\frac{\mathbb{E}\left[L^{(n)}\right]}{n} \to \mathbb{E}\left[W_Q\right],$$

since the number of items in the system, $\mathbb{E}\left[L^{(n)}\right]$, becomes ever larger as n increases, but the service time of one item is $1/n$, which becomes ever smaller. Thus, taking the limit $n \to \infty$ in (2.72) we obtain

$$\frac{\mathbb{E}\left[L^{(n)}\right]}{n} \to \frac{\rho}{1-\rho}\frac{1+C_s^2}{2}\mathbb{E}[S].$$

This is the same as Eq. (2.59)!

Exercise 2.19.1 In Eq. (2.71) we define $\mathbb{P}\left\{X^{(n)} = i\right\} = \mathbb{P}\{S \in (i/n,(i+1)/n]\}$ and not

$$\mathbb{P}\left\{X^{(n)} = i\right\} = \mathbb{P}\{S \in [i/n,(i+1)/n]\},$$

i.e., we use a half open interval $(,]$ rather than the closed interval $[,]$. Why is that?

Exercise 2.19.2 Why do we consider $\mathbb{E}\left[L^{(n)}\right]/n \to \mathbb{E}\left[W_Q\right]$, and not $\mathbb{E}\left[L^{(n)}\right]/n \to \mathbb{E}[W]$?

2.20 G/G/1 and G/G/c Queue: Approximations

In manufacturing settings it is quite often the case that the arrival process at a station is not Poisson. For instance, if processing times at a station are nearly constant, and the jobs of this station are sent to a second station for further processing, the inter-arrival times at the second station must be more or less equal. Hence, in this case, the SCV of the arrivals at the second station $C_{a,2}^2$ is most probably smaller than 1. As a consequence the Pollackzek-Khintchine formula for the $M/G/1$-queue can no longer be reliably used to compute the average waiting times. As a second, trivial case, if the inter-arrival times of jobs are 1 hour always and service times 59 minutes always, there simply cannot be queue. Thus, the $M/G/1$ waiting time formula cannot, and should not, be applied to approximate the average waiting time of the $G/G/1$ queue.

There is no formula as yet by which the average waiting times for the $G/G/1$ queue can be computed; only approximations are available. One such simple and robust approximation is based on the following observation. Recall the waiting time in queue for the $M/M/1$ queue:

$$\mathbb{E}\left[W_Q(M/M/1)\right] = \frac{\rho}{1-\rho}\mathbb{E}[S] = \frac{1_a+1_b}{2}\frac{\rho}{1-\rho}\mathbb{E}[S],$$

where we label the number 1 with a and b. When generalizing this result to the $M/G/1$ queue we get

$$\mathbb{E}\left[W_Q(M/G/1)\right] = \frac{1_a+C_s^2}{2}\frac{\rho}{1-\rho}\mathbb{E}[S],$$

Thus, 1_b in the expression for the $M/M/1$ queue is replaced by C_s^2 in the expression for the $M/G/1$ queue. As a second generalization, Kingman proposed to replace 1_a in this formula by the SCV of the inter-arrival times

$$C_a^2 = \frac{\mathbb{V}[X]}{(\mathbb{E}[X])^2}$$

resulting in

$$\mathbb{E}\left[W_Q(G/G/1)\right] \approx \frac{C_a^2 + C_s^2}{2} \frac{\rho}{1-\rho} \mathbb{E}[S].$$

This formula is reasonable accurate; for more, related, expressions we refer to Bolch et al. [2006] and Hall [1991]. With Little's law we can compute $\mathbb{E}\left[L_Q\right]$ from the above. Moreover, $\mathbb{E}[W] = \mathbb{E}\left[W_Q\right] + \mathbb{E}[S]$, and so on.

It is crucial to memorize the *scaling* relations that can be seen from the $G/G/1$ waiting time formula. Roughly:

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

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

3. $\mathbb{E}\left[W_Q\right] \sim \mathbb{E}[S]$. Thus, working in smaller job sizes reduces the waiting time also. The average queue length does not decrease by working with smaller batches, but jobs are more 'uniformly spread' over the queue. This effect lies behind the idea to do 'lot-splitting', i.e., rather than process large jobs, split jobs into multiple small jobs (assuming that setup times are negligible), so that the waiting time per job can be reduced.

These insights prove very useful when trying to reduce waiting times in any practical situation. First try to reduce the load (by blocking demand or increasing the capacity), then try to reduce the variability (e.g., by planning the arrival times of jobs), and finally, attempt to split jobs into multiple smaller jobs and use the resulting freedom to reschedule jobs in the queue.

For the $G/G/c$ queue we also only have an approximation:

$$\mathbb{E}\left[W_Q\right] = \frac{C_a^2 + C_e^2}{2} \frac{\rho^{\sqrt{2(c+1)}-1}}{c(1-\rho)} \mathbb{E}[S]. \tag{2.73}$$

Even though the above results are only approximate, they prove to be exceedingly useful when designing queueing systems and analyzing the effect of certain changes, in particular changes in capacity, variability and service times.

Clearly, Kingman's equation requires an estimate of the SCV C_a^2 of the inter-arrival times and the SCV C_s^2 of the service times. Now it is not always easy in practice to determine the actual service time distribution, one reason being that service times are often only estimated by a planner, but not actually measured. Similarly, the actual arrival moments of jobs are often not registered, mostly just the date or the hour, perhaps, that a customer arrived. Hence, it is often not possible to estimate C_a^2 and C_s^2 from information that is available. However, when for instance the number of arrivals per day have been logged for some time, i.e., we know $\{a_n, n = 1, \ldots, N\}$ for some N, we can this information instead of the the inter-arrival times $\{X_k\}$, to obtain insight into the relevant SCVs. Here we present a relation to compute C_a^2 from $\{a_n\}$; of course it can be applied to C_s^2 as well.

Theorem 2.20.1. The SCV of the inter-arrival times can be estimated with the following formula

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

where

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

in words, $\tilde{\lambda}$ is the average number of arrivals per period, e.g., per day, and $\tilde{\sigma}^2$ is the variance of the number of arrivals per period.

Proof. The proof is based on an argument in Cox [1962]. We use quite a bit of the notation developed in Section 2.7. Let $\{A(t), t \geq 0\}$ be the number of arrivals that occur up to (and including) time t. We assume that $\{A(t)\}$ is a renewal process such that the inter-arrival times $\{X_k, k = 1,2,\ldots\}$ with $X_k = A_k - A_{k-1}$, are i.i.d. with mean $1/\lambda$ and standard deviation σ. (Observe that σ is not the same as $\tilde{\sigma}$ above.) Note that C_a^2 is defined in terms of λ and σ as:

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

Next, let A_k be the arrival time of the kth arrival. The following useful relation between $A(t)$ and A_k enables us to prove our result (recall that we uses a similar relation in the derivation of the Poisson process):

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

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

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

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

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

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

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

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

Define for ease

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

We can use the above relation between the distributions of $A(t)$ and A_k to see that $\mathbb{P}\{A_k > t_k\} = \mathbb{P}\{A(t_k) < k\}$. With this we get,

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

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

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

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

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

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

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

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

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

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

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

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

Exercise 2.20.1 Show that (2.73) reduces to the result known for the $M/M/1$ and $M/G/1$ queues.

Exercise 2.20.2 Is Eq. (2.52) also valid for the $G/G/1$ queue? Why (not)?

Exercise 2.20.3 Consider a queue n servers, with generally distributed inter-arrival times, generally distributed service times, and the system can contain at most K customers, i.e, the $G/G/n/K$ queue. Let λ be the arrival rate, μ the service rate, β the long-run fraction of customers lost, and ρ the average number of busy/occupied servers Show that

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

Exercise 2.20.4

1. Consider a single-server queue at which every minute a customer arrives, precisely at the first second. Each customer requires precisely 50 seconds of service. What are ρ, $\mathbb{E}[L]$, C_a^2, and C_S^2?
2. Consider the same single-server system, 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?

Exercise 2.20.5 For the $G/G/1$ queue, prove that the fraction of jobs that see n jobs in the system is the same as the the fraction of departures that leave n jobs behind. What condition have you used to prove this?

Exercise 2.20.6 (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.

1. Assuming a single server, estimate $\mathbb{E}\left[W_Q\right]$ with the $G/G/1$ waiting time formula.
2. Compare this result to an $M/G/1$ system with arrival rate 10 per hour and the same service time distribution. Explain why your previous answer is smaller.

Exercise 2.20.7 What would be the difference between a multi-server queue and a single-server queue with a fast server? We can use the formulas for the $M/M/1$ queue and the $M/M/c$ queue to obtain some basic understanding of the difference. To this end, suppose we have an $M/M/3$ queue, with arrival rate $\lambda = 5$ per day and $\mu = 2$ per server, and we compare it to an $M/M/1$ with the same arrival rate but with a service rate of $\mu = 3 \cdot 2 = 6$.

1. Check that the service time of the slow server is three times as long as the service time of the fast server.
2. Estimate $\mathbb{E}\left[W_Q\right]$ for the $M/M/3$ queue, and compare this to $\mathbb{E}\left[W_Q\right]$ for the fast $M/M/1$ case.
3. Compare the sojourn times $\mathbb{E}[W]$ for both systems.
4. Implement the formulas in excel, or R, or some other computer environment, and make plots of $\mathbb{E}[W]$ as a function of the number of servers c.
5. When is it OK to approximate the $M/M/c$ queue by an $M/M/1$ queue with a fast server?

2.21 Service Interruptions

No notes; only questions.

Exercise 2.21.1 Determine the expectation and variance of $S_N = \sum_{i=1}^{N} X_i$ where N is a random variable (with finite second moment) and $\{X_i\}$ are i.i.d. as the generic random variable X.

Exercise 2.21.2

A small company receives orders at a rate λ. After a batch of B orders have been assembled, the company produces the batch. The average service time of an order is $1/\mu$. Between two batches the company has to re-adjust some of the machines before it can process another batch. A setup costs S time units.

1. Should the batch size be limited from above or below to ensure that the system remains stable?
2. How many orders arrive on average during a setup?
3. Use the above two points to determine a criterion on the batch size B.

Exercise 2.21.3 Zijm.Ex.1.11.1

Exercise 2.21.4 Zijm.Ex.1.11.2

Exercise 2.21.5 Zijm.Ex.1.11.3

Exercise 2.21.6 Zijm.Ex.1.11.4 and 1.11.5

Exercise 2.21.7 Zijm.Ex.1.11.4

Exercise 2.21.8 Zijm.Ex.1.11.6

Exercise 2.21.9 Zijm.Ex.1.11.7

2.22 Process Batching

No notes; only exercises.

Exercise 2.22.1. Zijm.Ex.1.12.1

Exercise 2.22.2. Zijm.Ex.1.12.2

Exercise 2.22.3. Zijm.Ex.1.12.3

3 Open Queueing Networks

Read the appropriate sections of Zijm's book.

3.1 Deterministic Queueing Networks

The simplest possible single-server queueing system is such that job interarrival times are deterministic and that all service times are equal. In such cases it is easy to determine the maximal throughput, i.e., the capacity of the queueing system. Clearly, if the service time is t_1, then the service rate is $r_1 = 1/t_1$. We can also determine the minimal amount of jobs required to keep the server busy, hence achieve maximal throughput. We call this the *critical work in progress (WIP)* and denote it by W_0. The minimal amount of time needed to move from the start of the system to the end, also called *raw processing time* is denoted by T_0. With Little's law we can relate these two concepts for the single-server example. If the arrival rate is equal to the service rate, then $\lambda = r_1$, so that

$$W_0 = \lambda T_0 = r_1 T_0.$$

Since, in the single server case $T_0 = t_1$, we see that $W_0 = r_1 t_1 = t_1^{-1} t_1 = 1$. This result is evident: to get maximal throughput, it is necessary that the server is always busy, hence it must be that $W_0 \geq 1$. Given that we deal with a $D/D/1$, by assumption, we can tune the arrivals such that whenever a job leaves, a new job arrives.

When we deal with *networks*, determining W_0 and T_0 is not so simple anymore. These quantities, however, are important to know. It is not possible to organize the network such that jobs spend less time than T_0 in the network. The raw processing time is the sum of the required processing times and does not include, by assumption, any queueing times. The critical WIP W_0 is important, because if the network contains less jobs than this amount, throughput will suffer. Stated in other terms, there not sufficient jobs in the network to keep bottleneck machines fully loaded with work. For these reasons T_0 and W_0 act as lower bounds on the time jobs spend in the network and the amount of work required to achieve a certain throughput. Finally, it is important to know the bottleneck capacity r_b of the network: The maximal throughput of the network cannot exceed the bottleneck capacity. Thus, when accepting jobs, one needs to respect the capacity of the network.

Before we study queueing networks in stochastic settings, we should familiarize ourselves with the behavior of networks of queueing systems in the simplest setting possible. For this we need some further definitions. A queueing network can be represented as *graph*. *Stations* form the nodes in the graph, and edges between nodes represents the possibility that jobs that can move from one station to another. A simple example is a tandem network with two stations. Jobs arrive at station 1 and after service they move to station 2. After service at station 2, they leave the network. A station can contain one or more servers. For instance, a single-server station contains just one server. Finally, we assume that job service times are constant at all servers at all stations. In other words, service times need to be same from one machine/server

or station to another, but each server works at constant speed. The *utilization* of station i is the rate $\lambda_i t_i$, i.e., the rate at which work arrives times the (average) service time per job. The *bottleneck* station is the station with the *highest utilization*.

The problems below are meant to help you become acquainted with networking behavior. They appear much simpler than they are; I had much less trouble making them than solving them. In fact, to ensure that I got the answer right, I often tried to find two different ways to solve the same problem. Be warned!

Exercise 3.1.1 A production system consists of 2 separate stations. Jobs never go from one machine to another. The processing times are $(t_1, t_2) = (1, 2)$ hours. The arrival rates are $(\lambda_1, \lambda_2) = (1/3, 1/3)$ per hour. What is the fastest machine? Which machine is the bottleneck?

Exercise 3.1.2 A production network consists of 3 single-machine stations in tandem, i.e., in series, c.f. Figure 3.1. The processing times are $(2, 3, 2)$ hours respectively, i.e., constant and such that $t_1 = 2$ hours, $t_2 = 3$ hours and $t_3 = 2$ hours.

What are the critical WIP W_0, bottleneck capacity r_b and raw processing time T_0?

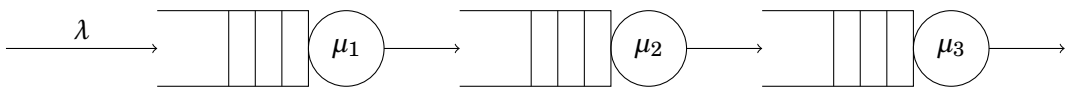


Figure 3.1: Three machines in tandem

Exercise 3.1.3 A production network consists of 3 single-machine stations in tandem with processing times $(2, 3, 2)$ hours. Suppose we modify this network by adding one extra machine to station 2, with processing time $t = 3$ hours. What are W_0, r_b and T_0?

Exercise 3.1.4 A production network consists of 3 single-machine stations in tandem with processing times $(2, 3, 2)$ hours. Suppose we modify this network by adding one extra machine to station 2 with processing time $t = 4$ hours. What are W_0, r_b and T_0?

Exercise 3.1.5 Why does the critical WIP change in the different cases above?

Exercise 3.1.6 A production network consists of 3 single-machine stations in tandem with processing times $(2, 3, 2)$ hours. Suppose we modify this network by adding one extra machine to station 2 with processing time $t = 1$ hour. What are W_0, r_b and T_0?

Exercise 3.1.7 In the previous question, it might seem that one machine at the second station is superfluous. Which one, and why? Would you remove this superfluous machine in a real production network?

Exercise 3.1.8 A production network consists of 3 single-machine stations in tandem with processing times $(2, 3, 2)$ hours. Suppose we modify this network by adding one extra machine to station 2 with processing time $t = 7$ hours. What are W_0, r_b and T_0?

Exercise 3.1.9 A production network consists of 3 single-machine stations in tandem with processing times $(2, 3, 2)$ hours. Suppose we modify this network such that 2/3 of the jobs *leave after the first* station, i.e., between station 1 and 2. What are W_0, r_b and T_0?

Exercise 3.1.10 A production network consists of 3 single-machine stations in tandem with processing times (2,3,2) hours. Suppose we modify this network such that 4/5 of the jobs *bypass the second* station, i.e., 1/5 of the jobs have the routing [1,2,3], but 4/5 have routing [1,3]. What are W_0, r_b and T_0?

Exercise 3.1.11 For the interested: production situations can be pretty complicated. Jobs can enter the network at different places, not just at station 1 as we considered here. Jobs may also need rework at one or more stations. One consequence of these aspects is that the slowest machines or stations need not be the bottlenecks. Can you find or develop a general algorithm by which we can compute T_0, W_0 and r_b for any network of reasonable size? (I have not tried this... yet.)

One way to approach this problem is by placing (imaginary) huge amounts of work in front of each station and just see how the network drains.

3.2 Open Single-Class Product-Form Networks

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

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

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

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

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

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

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

where we recognize the visit ratios.

Exercise 3.2.1 The aim of the question is the determine the distribution of the inter-departure times of the $M/M/1$ queue. Once we have characterized this distribution, we can characterize the distribution of the arrival process of a queueing system immediately downstream of the queueing system under consideration.

We chop up the problem in small steps.

1. Is the output rate of the $M/M/1$ queue λ or μ?
2. Why is $\mu e^{-\mu t}$ not a reasonable density for the inter-departure times?
3. Formulate a better guess for the inter-departure times.

4. Observe that when a customer arrives, the server can be either idle I or busy B. What is $\mathbb{P}\{I\}$?
5. If customer n, say, finds the system empty, what will be the expected duration of the inter-departure time between D_{n-1} and D_n?
6. What is the density of $D_n - D_{n-1}$ when job n sees an empty system?
7. What is $\mathbb{P}\{B\}$, i.e., the probabily that a job arrives at a busy station?
8. What is the density of the inter-departure time if the server is busy?
9. What is the final result for the inter-departure time?
10. It can also be shown that the inter-departure times are independent. What is the implication of the last two statements?

Exercise 3.2.2 Zijm.Ex.2.2.1

Exercise 3.2.3 Zijm.Ex.2.2.2

Exercise 3.2.4 Zijm.Ex.2.2.3

Exercise 3.2.5 Zijm.Ex.2.2.4

Exercise 3.2.6 Zijm.Ex.2.2.5. The problem is not entirely correctly formulated. It should be, if for at least one i, $\sum_{j=1}^{M} P_{ij} < 1 \ldots$

Exercise 3.2.7 Zijm.Ex.2.2.6

Exercise 3.2.8 Show that Zijm.Eq.2.13 and 2.14 can be written as

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

Exercise 3.2.9 Zijm.Ex.2.2.7

Exercise 3.2.10 Zijm.Ex.2.2.8

3.3 Tandem queues

No notes; only exercises.

Exercise 3.3.1 Consider two stations in tandem. The service process at both stations is subject to variability. Suppose you have some money to spend on reducing the variability of either of the two machines. Which one is the better one to spend it on? You should realize the relevancy of modeling for this problem. Due to the lack of money you can only realize one of the alternatives; hence there is no way of finding out what would have happened if you would have made the other choice.

Exercise 3.3.2 (Hall 5.22). At a large hotel, taxi cabs arrive at a rate of 15 per hour, and parties of riders arrive at the rate of 12 per hour. Whenever taxicabs are waiting, riders are served immediately upon arrival. Whenever riders are waiting, taxicabs are loaded immediately upon arrival. A maximum of three cabs can wait at a time (other cabs must go elsewhere).

1. Let p_{ij} be the steady-state probability of there being i parties of riders and j taxicabs waiting at the hotel. Write the state transition equation for the system.
2. Calculate the expected number of cabs waiting and the expected number of parties waiting.
3. Calculate the expected waiting time for cabs and the expected waiting for parties. (For cabs, compute the average among those that do not go elsewhere.)
4. In words, what would be the impact of allowing four cabs to wait at a time?

3.4 General Job Shop Manufacturing Systems

Perhaps the term 'class' needs some clarification. A network with a single class of jobs means that the all jobs have the same service distribution. In multi-class networks jobs of different classes typically have different service distributions. Note that we do not discuss multi-class queueing networks in this course.

Exercise 3.4.1 It is mentioned in the text that there are $5M + M^2$ input parameters required to characterize an M station network of $G/G/1$ queues. What are these parameters?

Exercise 3.4.2 Consider a tandem of single server work stations. If you would be given some money to reduce the variance of the service times of one machine. Which machine would you prefer?

Exercise 3.4.3 Assume that in a network of five stations the last station is the bottleneck. Under what conditions would you invest in this last work station?

4 Closed Queueing Networks

4.1 Gordon-Newell Networks

The formula with the visit ratios should be like this:

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

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

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

It is perhaps the easiest to start with studying the MVA algorithm, then the MDA algorithm and finish with the convolution algorithm.

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

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

Exercise 4.1.1 Compute the visit ratios for a network with three stations such that all jobs from station 0 move to station 1, from station 1 all move to station 2, and from station 2 half of the jobs move to station 0 and the other half to station 1.

Exercise 4.1.2 What if P has an eigenvalue α whose modulus $|\alpha| >$? What if for all eigenvalues α_i we have that $|\alpha_i| < 1$?

Exercise 4.1.3 Zijm.Ex.3.1.1

Exercise 4.1.4 Zijm.Ex.3.1.2

Exercise 4.1.5 Zijm.Ex.3.1.3

Exercise 4.1.6 Relate Zijm.Eq.3.3 to the form of the steady-state distribution of the number of jobs in an $M/M/c$ queue.

Exercise 4.1.7 Zijm.Ex.3.1.4

Exercise 4.1.8 Zijm.Ex.3.1.5

4.2 The Convolution Algorithm

In view of the later algorithm, it is more consistent to write $p_i(k|N) = \pi(n_i = k)$ for the marginal probability that station i contains k jobs, under the assumption that the network contains N jobs.

Assume we have computed $p_i(k|N)$ with Eq.3.15, i.e., with the recursion

$$p_i(k|N) = f_i(k)\frac{G(M \setminus \{i\}, N-k)}{G(M,N)}.$$

We can then compute the rest of the performance measures as

$$\begin{aligned}
\mathrm{TH}_i &= \mu_i \sum_{k=1}^{N} p_i(k|N), \\
\mathbb{E}[W] &= \frac{N}{TH_0}, \\
\mathbb{E}[L]_i &= \sum_{k=1} k p_i(k|N), \\
\mathbb{E}[W]_i &= \mathbb{E}[L]_i / V_i, \text{ by Little's law.}
\end{aligned}$$

The rest of the equations in Section 3.1.2 are interesting, but not necessary for an implementation in computer code.

Exercise 4.2.1 Explain Zijm.Eq.3.13.

Exercise 4.2.2 TBD. Relate the following to the arrival theorem below. The derivation of Zijm.Eq.3.8 is particularly interesting. Rewrite Zijm.Eq.3.5 as

$$\pi_{M,N}(N_i = k) = f_i(k) \frac{G(M \setminus \{i\}, N-k)}{G(M,N)},$$

to denote, *for a network with M stations and N jobs*, the (marginal) stationary distribution that station i contains k jobs. With this notation

$$\begin{aligned}
\mathrm{TH}_i &= \frac{V_i}{G(M,N)} \sum_{k=1}^{N} f_i(k-1) G(M \setminus \{i\}, N-k) \\
&= \frac{V_i}{G(M,N)} \sum_{k=0}^{N-1} f_i(k-1) G(M \setminus \{i\}, N-k-1) \\
&= V_i \sum_{k=0}^{N-1} f_i(k-1) \frac{G(M \setminus \{i\}, N-k-1)}{G(M,N)} \\
&= \sum_{k=0}^{N-1} (V_i \pi_{M,N-1}(N_i = k-1)) \\
&= V_i \frac{G(M,N-1)}{G(M,N)}.
\end{aligned}$$

The second to last equation is the most interesting in fact. This says that the throughput of station i is (the sum of) the visit ratio V_i times the stationary probabilities of a system with one job less, i.e., $N-1$, and that this job is taken away from station i, i.e., $N_i = k-1$.

Exercise 4.2.3 Zijm.Ex.3.1.6

Exercise 4.2.4 Zijm.Ex.3.1.7

Exercise 4.2.5 Zijm.Ex.3.1.8

Exercise 4.2.6 Zijm.Ex.3.1.9. This question forces you to really trace down all relevant definitions. Please try to do it yourself first. It is harder than you might think initially.

Exercise 4.2.7 Zijm.Ex.3.1.10. Assume that the visit ratios are 1.

Exercise 4.2.8 Zijm.Ex.3.1.11

Exercise 4.2.9 Zijm.Ex.3.1.12

4.3 MVA Algorithm

Exercise 4.3.1 Consider two stations in tandem, stations 0 and 1. The service times are $\mathbb{E}[S_0] = 2 = 1/\mu_0$ and $\mathbb{E}[S_1] = 3 = 1/\mu_1$ hours. The routing matrix is

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

Apply the MVA algorith to this case.

Exercise 4.3.2 Zijm.Ex.3.1.13. Assume that all stations have just one server.

Exercise 4.3.3 Zijm.Ex.3.1.14

Exercise 4.3.4 Implement the MVA algorithm in your preferred computer language and make Figure 3.2.

4.4 MDA Algorithm

A relatively easy way to obtain insight into the procedure is by comparing this case to the single-server single station case. Recall that, for the latter case,

$$\mathbb{E}\left[W_Q\right] = \rho\,\mathbb{E}[S_r \,|\, S_r > 0] + \mathbb{E}\left[L_Q\right]\mathbb{E}[S].$$

The first term is the probability that the server is found busy by an arriving job times the expected remaining service time of the job in service; the second is the expected time to clear the queue.

For the closed queueing network, the probability to find k at station j is $p_j(k|n-1)$, where we use the arrival theorem that states when n jobs are present in the network, a 'jumping' job sees the stationary distribution of the same network but with one job less.

To wait in queue at station j it is necessary that $k \geq c_j$, for otherwise service can start right at the arrival moment. Thus, the probability that all servers are busy is

$$\sum_{k=c_j}^{n-1} p_j(k|n-1).$$

(Compare the first ρ in the above equation for $\mathbb{E}\left[W_Q\right]$). The service of the queue can start once the first job currently in service leaves. This takes $1/c\mu_j$. (Compare this to $\mathbb{E}[S_r \,|\, S_r > 0]$.)

The expected to clear the queue is $\mathbb{E}[L_{Q,j}]$ divided by the rate at which jobs are served. Thus, this time must be $\mathbb{E}[L_{Q,j}]/c\mu_j$. Now,

$$\mathbb{E}[L_{Q,j}] = \sum_{k=c_j}^{n-1} (k-c_j) p_j(k|n-1).$$

All in all, the total average time at station j is the average time in queue plus the average service time of the job itself:

$$\mathbb{E}[W_j(n)] = \frac{1}{c_j\mu_j} \sum_{k=c_j}^{n-1} p_j(k|n-1) + \frac{1}{c_j\mu_j} \sum_{k=c_j}^{n-1} (k-c_j) p_j(k|n-1) + \frac{1}{\mu_j},$$

Up- and down-crossing of level k gives that

$$\mathrm{TH}_j(n) p(k-1|n-1) = \min(k, c_j) p(k|n). \tag{4.1}$$

Exercise 4.4.1 Apply the MDA algorithm to the network discussed in Question 1, but now assume that station 1 has 2 servers.

Exercise 4.4.2 Zijm.Ex.3.1.15

Exercise 4.4.3 Consider Example 3.5. Suppose you would add an extra machine to station 0. What will become the throughput and the average number of jobs at each station? Use the Marginal Distribution Analysis Algorithm. Plot also the distribution of the number of jobs in stations 0 and 1.

5 Hints

Hint 2.1.1 For the second child, condition on the event that the first does not chose the right number.

Hint 2.1.3 Condition on the event $X > t$.

Hint 2.1.9 Check the result for $i = 1$ by filling in $i = 1$ (just to be sure that you have read the formula right), and compare the result to the exponential density. Then write $A_i = \sum_{i=1}^{k} X_k$, and compute the moment generating function for A_i and use that the inter-arrival times X_i are independent. Use the moment generating function of X_i.

Hint 2.1.10 Use that $\int_0^\infty (\lambda t)^i e^{-\lambda t}\,\mathrm{d}t = \frac{i!}{\lambda}$. Another way would be to use that, once you have the moment generating function of some random variable X, $\mathbb{E}[X] = \frac{\mathrm{d}}{\mathrm{d}t} M(t)|_{t=0}$.

Hint 2.1.11 Use that if $Z = \min\{X, Y\} > x$ that then it must be that $X > x$ and $Y > x$. Then use independence of X and Y.

Hint 2.1.12 Define the joint distribution of A and S and carry out the computations, or use conditioning, or use the result of the previous exercise.

Hint 2.2.3 First find p, n, λ and t are such that the rate at which event occur in both processes are the same. Then consider the binomial distribution and use the standard limit $(1 - x/n)^n \to e^{-x}$ as $n \to \infty$.

Hint 2.2.4 Realize that $\mathbb{P}\{N(t) = k\} = \mathbb{P}\{A_k \le t\} - \mathbb{P}\{A_{k+1} \le t\}$.

Hint 2.2.5 Think about the meaning of the formula $\mathbb{P}\{N(h) = n \,|\, N(0) = n\}$. It is a conditional probability that should be read like this: given that up to time 0 we have seen n arrivals (i.e, $N(0) = n$), what is the probability that just a little later (h) the number of arrivals is still n, i.e, $N(h) = n$? Then use the definition of the Poisson distribtuion to compute this probability. Finally, use Taylor's expension of e^x to see that $e^x = 1 + x + o(x)$ for $|x| << 1$. Furthermore, use that $\sum_{i=2}^{\infty} x^i/i! = \sum_{i=0}^{\infty} x^i/i! - x - 1 = e^x - x - 1$.

Hint 2.2.7 Use a conditioning argument or use moment generating functions. In particular, for conditioning, use that $\mathbb{P}\{AB\} = \mathbb{P}\{A \,|\, B\}\,\mathbb{P}\{B\}$. More generally, if the set A can be split into disjoint sets B_i, i.e, $A = \bigcup_{i=1}^{n} B_i$, then

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

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

Hint 2.2.8 Suppose we write $N(t) = N_a(t)+N_s(t)$. Then $\mathbb{P}\{N_a(h) = 1, N_s(h) = 0 | N_a(h)+N_s(h) = 1\}$ is the probability that $N_a(h) = 1$ and $N_s(h) = 0$ *given* that $N(t) = 1$. In other words, the question is find out that, given one of the two processes 'fired', what is the probability that N_a was the one that 'fired'.

Hint 2.2.10 Condition on the total number of arrivals $N(t) = m$ up to time t. Realize that the probability that a job is of type 1 is Bernoulli distributed, hence when you consider m jobs in total, the number of type 1 jobs is binomially distributed.

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

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

Now choose practical sets B_i.

You might also consider the random variable

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

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

Hint 2.2.11 You might use generating functions here.

Hint 2.4.18 Formuale the decision variables/controls, the objective and the constraints.

Hint 2.5.2 Use that $W_{Q,1} = [W_{Q,0} + S_1 - X_1]^+$ and likewise for $W_{Q,2}$.

This first year probability. It should be elementary for you. Ensure you can do it.

Hint 2.5.3 This should be trivial, of course, but I didn't find it easy... In fact, I found it a major headache, hence it is good to try yourself before looking at the answer. Check your probability book if you don't know what to do.

Hint 2.5.5 Compare this to the definition in (2.9).

Hint 2.6.1 When are the arrivals in slot k available for service in either of the systems?

Hint 2.6.2 Note first that from the expression for Z_k, $a_k - c_k = Z_k - Z_{k-1}$. Use this to get $Q_k = [Q_{k-1} + Z_k - Z_{k-1}]^+$. Subtract Z_k, use recursion and use subsequently,

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

Hint 2.6.3 It is actually not hard, even though the expression looks hard. Use condition to see that $\mathbb{P}_{Z(t)=n}(=)\,\mathbb{P}\{N_\mu(t) - N_\lambda(t) = m - n\}$. Then write out the definitions of the two Poisson distributions. Assemble terms. Then fiddle a bit with the terms to get $(t\sqrt{\lambda\mu})$.

Hint 2.6.4 Consider $Z(t) = Z(0) + N_\lambda(t) - N_\mu(t)$ at the embedded set of epochs at which either N_λ or N_μ changes. What is the probablity that N_λ fires, given that one of the two Poisson processes fires? Relate this to $\{a_k\}$ and $\{c_k\}$.

Hint 2.7.8 Here are some questions to help you interpret this formulation.

1. What are the decision variables for this problem? In other words, what are the 'things' we can control/change?
2. What are the interpretations of $k_A t_A$, and $S + k_B t_B$?
3. What is the meaning of the first constraint? Realize that T represents one production cycle. After the completion of one such cycle, we start another cycle. Hence, the start of every cycle can be seen as a restart of the entire system.
4. What is the meaning of the other two constraints?
5. Why do we minimize the cycle time T?
6. Solve for k_A and k_B in terms of S, λ_A, λ_B and t_A, t_B.
7. Generalize this to m job classes and such that the cleaning time between jobs of class i and j is given by S_{ij}. (Thus, the setup times are sequence dependent.)

Hint 2.8.1 Consider a queueing system with constant service and interarrival times.

Hint 2.8.3 Why is (2.26) not the same as the number of batches that see a queue length less than m?

Hint 2.9.7 For the acronym, observe that the service times and interarrival are *D*eterministic and there is one server. For the computation of $Y(n,t)$, make make a plot of $L(s)$ as a function of time for $n = 1$.

Hint 2.10.3 Realize that the inventory level $I(t)$ at time t can be modeled as $I(t) = r + 1 - L(t)$, where L is the number of jobs in an $M/M/1$ queue.

Hint 2.11.1 Find suitable expressions for $\lambda(n)$ and $\mu(n)$ and use the level-crossing equations $\lambda(n)p(n) = \mu(n+1)p(n+1)$.

Hint 2.11.2 Use $\lambda(n)p(n) = \mu(n+1)p(n+1)$ and find suitable expressions for $\lambda(n)$ and $\mu(n+1)$.

Hint 2.11.4 Use $\lambda(n)p(n) = \mu(n+1)p(n+1) = \min\{c, n+1\}\mu p(n+1)$.

Hint 2.11.6 Use $\lambda(n)p(n) = \mu(n+1)p(n+1)$, and realize that for this case $\lambda(n) = (N-n)\lambda$ and $\mu(n) = \mu$.

Hint 2.12.4 Think about the construction of the $M/M/1$ queue as a random walk, see Section 2.6.

Hint 2.13.1 Think about the data that is given in either situation.

Hint 2.13.2 Make a drawing of $A(t)$ and $D(t)$ until time T, i.e., the first time the system is empty.

Hint 2.14.1 This requires some extra definitions, similar to $A(n,t)$ as defined in (2.30a).

Hint 2.14.12 Use the level-crossing equations of the $M(n)/M(n)/1$ queue.

Hint 2.14.13 This is a queueing system with loss, in particular the $M/M/1/1+2$ queue.

Hint 2.14.19 Interpret each component at the right hand side of the equation and generalize it to a multi-server queueing system. Then use Exercise 11.

Hint 2.15.2 This exercise is just meant to become familiar with the notation.

Hint 2.15.4 Realize that the inventory process $I(t)$ behaves as $I(t) = r - L(t)$ where $L(t)$ is a suitable queueing process.

Hint 2.15.5 Write $\sum_{n=0}^{\infty} G(n) = \sum_{n=0}^{\infty} \sum_{i=n+1}^{\infty} \mathbb{P}\{B = i\}$ and then do the algebra.

Hint 2.15.6 $\sum_{i=0}^{\infty} iG(i) = \sum_{n=0}^{\infty} \mathbb{P}\{B = n\} \sum_{i=0}^{\infty} i1\{n \geq i+1\}$.

Hint 2.15.7 $\mathbb{E}[S] = \int_0^{\infty} x\,\mathrm{d}F = \int_0^{\infty} \int_0^{\infty} 1_{y \leq x}\,\mathrm{d}y\,\mathrm{d}F(x)$.

Hint 2.15.8 $\int_0^{\infty} yG(y)\,\mathrm{d}y \int_0^{\infty} y \int_0^{\infty} 1\{y \leq x\} f(x)\,\mathrm{d}x\,\mathrm{d}y$.

Hint 2.15.11 What is the distribution of the batch size B for the $M/M/1$ queue?

Hint 2.17.4 Substitute the recursion, and carry on with the algebra. We urge the reader to try it, its good (necessary) practice.

Hint 2.17.5 This exercise tests your creativity and modeling skills, it is not analytically difficult. The best approach to problems like this is to try some simple cases first. For instance, consider the case with batch sizes of 1 first, then batches of sizes 1 or 2, and so on. If system contains K jobs, which batches can be accepted? If the system contains $K-3$, say, which batch sizes can be accepted, which will be refused, under which acceptance policy?

Hint 2.17.6 This is an important problem, it helps to check (numerically) the algebraic results we have been deriving up to now. Implementing the recursion is not hard, just try and see how far you get.

Hint 2.18.2 Realize that if $L(D_{k-1}) = 0$, job $k-1$ leaves behind an empty system. Thus, before job k can leave, it has to arrive. In other words, $D_{k-1} < A_k$. Since job k arrives to an empty system, his service starts right away, to that the time between A_k and D_k is equal to the service time of job k.

Hint 2.18.3 If s is deterministic, the number of arrival during a fixed period of time with length s must be Poisson distributed.

Hint 2.18.8 Define shorthands such as $\alpha = \lambda/(\lambda+\mu)$, so that $1-\alpha = \mu/(\lambda+\mu)$, and $\alpha/(1-\alpha) = \lambda/\mu = \rho$. Then, with the previous exercise that $f(n) = \alpha^n(1-\alpha)$ and $G(n) = \alpha^{n+1}$. (I did not get this result for 'free'. Indeed, it's just algebra, but please try to derive this yourself. Its good to hone your computational skills.)

Hint 2.19.2 What is the expected service time of one unit of a job in the $M^X/M/1$ queue in the limiting case?

Hint 2.20.1 Take $c = 1$.

Hint 3.1.3 Is the new machine placed in parallel or in series?

Hint 3.1.4 Check your reasoning very carefully here. It is easy to do it wrong.

Hint 3.2.1 The final result is $f_D(t) = f_{X+S}(t)\mathbb{P}\{I\} + f_S\,\mathbb{P}\{B\} = (1-\rho)f_{X+S}(t) + \rho\mu e^{-\mu t}$. Now use the above to simplify and see that $f_D(t) = \lambda e^{-\lambda t}$.

Hint 3.3.2 Try to adapt the ideas behind Figure 2.2 of Zijm to this case.

Hint 3.4.1 Check the formulas in the synthesis carefully.

Hint 4.1.8 First write down all different states, and then use Zijm.Eq.3.2 and 3.3.

Hint 4.2.7 Compare Zijm.Question 3.1.5.

Hint 4.3.1 Start with $n = 1$, then consider $n = 2$ and so on.

Hint 4.4.1 Just follow the description of the algorithm of the book of Zijm. I expect that you have read the solution of Question 1.

6 Solutions

Solution 2.1.1 Use the definition of conditional probability ($\mathbb{P}\{A|B\} = \mathbb{P}\{AB\}/\mathbb{P}\{B\}$, provided $\mathbb{P}\{B\} > 0$)

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

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

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

Solution 2.1.2 Write for $F(x) = \mathbb{P}\{X \leq x\}$ and $f(x) = \mathrm{d}F(x)/\mathrm{d}x$ for the density of F.

$$
\begin{aligned}
\mathbb{E}[h(X)] &= \int_0^{10} \mathbb{E}[h(X)\,|\,X = x]\,\mathbb{P}\{X \in \mathrm{d}x\} \\
&= \int_0^{10} \mathbb{E}[h(x)\,|\,X = x]\,\mathrm{d}F(x) \\
&= \int_0^{10} \mathbb{E}[h(x)\,|\,X = x]F(\mathrm{d}x) \\
&= \int_0^{10} \mathbb{E}[h(x)\,|\,X = x]f(x)\mathrm{d}x \\
&= \int_0^{10} h(x)\frac{\mathrm{d}x}{10}.
\end{aligned}
$$

Here we introduce some notation that is commonly used in the probabity literature to indicate the same conceptual idea, i.e, $\mathbb{P}\{X \in \mathrm{d}x\} = \mathrm{d}F(x) = F(\mathrm{d}x) = f(x)\mathrm{d}x$, where the last equality follows from the fact that F has a density f everywhere on $[0,10]$.

The concept of conditional expectation is of fundamental importance in probability theory. Any *good* probability book defines this concept as a random variable measurable with respect to some σ-algebra. In this course we will not deal with this elegant idea, due to lack of time.

Solution 2.1.3 This is easy, but be sure you can do it.

To see that an exponentially distributed is memoryless, use the definition of conditional probability ($\mathbb{P}\{A|B\} = \mathbb{P}\{AB\}/\mathbb{P}\{B\}$, provided $\mathbb{P}\{B\} > 0$):

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

Solution 2.1.4

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

Solution 2.1.5

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

Solution 2.1.6 By the previous problems, $\mathbb{E}\left[X^2\right] = 2/\lambda^2$ and $\mathbb{E}[X] = \lambda$.

Solution 2.1.7 By the previous problems, $\mathbb{V}[X] = 1/\lambda^2$ and $\mathbb{E}[X] = 1/\lambda$.

Solution 2.1.8

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

Solution 2.1.9 One way to find the distribution of A_i is by using the moment generating function $M_{A_i}(t) = \mathbb{E}\left[e^{tA_i}\right]$ of A_i. Let X_i be the inter-arrival time between customers i and $i-1$, and $M_X(t)$ the associated moment generating function. Using the i.i.d. property of the $\{X_i\}$,

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

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

Solution 2.1.10

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

Thus,

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

where we used the hint.

What if we would use the moment generating function?

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

Of course, $\mathbb{E}[A]_i = \mathbb{E}\left[\sum_{k=1}^i X_k\right] = i\,\mathbb{E}[X] = 1/\lambda$.

Solution 2.1.11 Use that X and Y are independent to get

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

Solution 2.1.12 There is more than one way to show that $\mathbb{P}\{A \le S\} = \lambda/(\lambda+\mu)$.

Method 1. (I admit that, although the simplest, least technical, method, I did not think of this right away. I am 'conditioned' to use conditioning...) Observe first that A and S, being exponentially distributed, both have a density. Moreover, as they are independent, we can

sensibly speak of the joint density $f_{A,S}(x,y) = f_A(x)f_S(y) = \lambda\mu e^{-\lambda x}e^{-\mu y}$. With this,

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

This argument is provided in the probability book you use in the first year.

Method 2. Applying a standard conditioning argument

$$\mathbb{P}\{A \le S\} = \int_0^\infty \mathbb{P}\{A \le S | S = s\}\,\mu e^{-\mu s}\,\mathrm{d}s.$$

Now, $\mathbb{P}\{A \le S | S = s\}$ is a conditional probability distribution. This is a bit of tricky object, but very useful once you get used to it. The tricky part is that $\mathbb{P}\{S = s\} = 0$. Therefore $\mathbb{P}\{A \le S \,|\, S = s\}$ cannot be defined as $\frac{\mathbb{P}\{A\le s, S=s\}}{\mathbb{P}\{S=s\}}$. However, if we proceed nonetheless and use the independence of S and A, we get

$$\mathbb{P}\{A \le S | S = s\} = \frac{\mathbb{P}\{A \le s, S = s\}}{\mathbb{P}\{S = s\}} = \frac{\mathbb{P}\{A \le s\}\,\mathbb{P}\{S = s\}}{\mathbb{P}\{S = s\}} = \mathbb{P}\{A \le s\}$$

and thus, indeed, $\mathbb{P}\{A \le S | S = s\} = \mathbb{P}\{A \le s\}$. Then,

$$\begin{aligned}
\mathbb{P}\{A \le S\} &= \int_0^\infty \mathbb{P}\{A \le S | S = s\}\,\mu e^{-\mu s}\,\mathrm{d}s \\
&= \int_0^\infty \mathbb{P}\{A \le s\}\,\mu e^{-\mu s}\,\mathrm{d}s \\
&= \int_0^\infty (1 - e^{-\lambda s})\mu e^{-\mu s}\,\mathrm{d}s
\end{aligned}$$

and we arrive at the integral we have seen above.

So we get the correct answer, but by the wrong method. How can we repair this? As a first step, let's not fix S to a set of measure zero, but let's assume that $S \in [s,t]$ for $s < t$. Then it follows that

$$1\{A \le s\}1\{S \in [s,t]\} \le 1\{A \le S\}1\{S \in [s,t]\} \le 1\{A \le t\}1\{S \in [s,t]\}$$

As a second step, using that $\mathbb{P}\{S \in [s,t]\} > 0$ if $s < t$ and the independence of A and S,

$$\begin{aligned}
\mathbb{P}\{A \le s\} &= \frac{\mathbb{P}\{A \le s\}\,\mathbb{P}\{S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}} = \frac{\mathbb{P}\{A \le s, S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}} \\
\mathbb{P}\{A \le t\} &= \frac{\mathbb{P}\{A \le t\}\,\mathbb{P}\{S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}} = \frac{\mathbb{P}\{A \le t, S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}}
\end{aligned}$$

Now with the result of the first step

$$\begin{aligned}\mathbb{P}\{A \le s\} &= \frac{\mathbb{P}\{A \le s, S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}} \\ &\le \frac{\mathbb{P}\{A \le S, S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}} \\ &= \mathbb{P}\{A \le S | S \in [s,t]\} \\ &\le \frac{\mathbb{P}\{A \le t, S \in [s,t]\}}{\mathbb{P}\{S \in [s,t]\}} \\ &= \mathbb{P}\{A \le t\}.\end{aligned}$$

Hence,

$$\mathbb{P}\{A \le s\} \le \mathbb{P}\{A \le S | S \in [s,t]\} \le \mathbb{P}\{A \le t\}.$$

Finally, taking the limit $t \downarrow s$, and defining $\mathbb{P}\{A \le S | S = s\} = \lim_{t \downarrow s} \mathbb{P}\{A \le S | S \in [s,t]\}$, it follows that

$$\mathbb{P}\{A \le s\} = \mathbb{P}\{A \le S | S = s\}$$

A more direct way to properly define $\mathbb{P}\{A \le S | S = s\}$ is as follows. For any y such that $f_S(y) > 0$, we can define the conditional probability density function of A, given that $S = s$, as

$$f_{A|S}(x|s) = \frac{f_{A,S}(x,s)}{f_Y(s)},$$

where, as before, $f_{A,S}(x,s)$ is the joint density of A and S. Now that the conditional probability density is defined, we can properly define

$$\mathbb{E}[A | S = s] = \int_0^\infty x f_{A|S}(x|s)\, \mathrm{d}x$$

and also

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

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

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

from which we get that

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

that is,

$$\mathbb{P}\{A \le S | S = s\} = \mathbb{E}[\mathbb{1}_{A\le S} | S = s] = 1 - e^{-\lambda s} = \mathbb{P}\{A \le s\}.$$

All of these problems can be put on solid ground by using measure theory. We do not pursue these matters any further, but trust on our intuition that all is well.

Solution 2.1.13 Let X be the processing (or service) time at the server, and X_i the service time of a type i job. Then,

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

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

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

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

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

we get

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

Interestingly, we see that even if $\mathbb{V}[X]_1 = \mathbb{V}[X]_2 = 0$, $\mathbb{V}[X] > 0$ if $\mathbb{E}[X_1] \neq \mathbb{E}[X_2]$. Bear this in mind; we will use these ideas later when we discuss the effects of failures on the variance of service times of jobs.

Solution 2.1.14 I used python to do the simulations. I include the code for you to study if you are interested, but you are free to skip it; it is not part of the core of the course.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_context("paper")

np.random.seed(3)


def superposition(a):
    A = np.cumsum(a, axis=1)
    A = np.sort(A.flatten())
    a = A[1:] - A[:-1]
    return a


n = 100 # simulation run length
x = np.arange(0, 8, 0.1)


def makePlot(A, N):
    plt.axis([0, 8, 0, 2])

    X = superposition(A)
    labda = N / A.mean()
    plt.hist(X, 20, normed=1, label='simulation')
    plt.plot(x, labda * np.exp(-labda * x), label="theory")
    plt.title("N = {}, n = {}".format(N, n))
    plt.legend()


plt.figure(figsize=(6, 2))
plt.subplot(131)
N = 1
A = np.random.uniform(4, 6, (N, n))
makePlot(A, N)

plt.subplot(132)
N = 3
A = np.random.uniform(4, 6, (N, n))
makePlot(A, N)

plt.subplot(133)
N = 10
A = np.random.uniform(4, 6, (N, n))
makePlot(A, N)

filename = "figures/uniform_N_{}_n_{}.pdf".format(N, n)
plt.savefig(filename)
plt.close()
```

```
# normal distribution

plt.figure(figsize=(6, 2))
plt.subplot(131)
N = 1
A = np.random.normal(5, 1, (N, n))
makePlot(A, N)

plt.subplot(132)
N = 3
A = np.random.normal(5, 1, (N, n))
makePlot(A, N)

plt.subplot(133)
N = 10
A = np.random.normal(5, 1, (N, n))
makePlot(A, N)

filename = "figures/normal_N_{}_n_{}.pdf".format(N, n)
plt.savefig(filename)
plt.close()
```

Solution 2.2.1

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

Solution 2.2.2 $N_n(t)$ is a binomially distributed random variable with parameters n and p. The maximum value of $N_n(t)$ is n. The random variable $N(t)$ models the number of failures that can occur during $[0,t]$. As such it is not necessarily bounded by n. Thus, $N_n(t)$ and $N(t)$ cannot represent the same random variable.

Solution 2.2.3 Now we like to relate $N_n(t)$ and $N(t)$. It is clear that we at least want the expectations to be the same, that is, $np = \lambda t$. This implies that

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

so that

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

To see that

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

use that

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

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

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

Solution 2.2.4 We want to show that

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

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

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

We are done.

Solution 2.2.5

1. $\mathbb{P}\{N(h) = n | N(0) = n\} = \mathbb{P}\{N(h) = 0\} = e^{-\lambda h}(\lambda h)^0/0! = e^{-\lambda h} = 1 - \lambda h + o(h)$, as follows from a standard argument in analysis that $e^{-\lambda h} = 1 - \lambda h + o(h)$ for h small.
2. $\mathbb{P}\{N(h) = n+1 | N(0) = n\} = \mathbb{P}\{N(h) = 1\} = e^{-\lambda h}(\lambda h)^1/1! = (1 - \lambda h + o(h))\lambda h = \lambda h - \lambda^2 h^2 + o(h) = \lambda h + o(h)$.
3. $$\begin{aligned}
\mathbb{P}\{N(h) \ge n+2 | N(0) = n\} &= \mathbb{P}\{N(h) \ge 2\} \\
&= e^{-\lambda h} \sum_{i=2}^{\infty} \frac{(\lambda h)^i}{i!} = e^{-\lambda h} \left(\sum_{i=0}^{\infty} \frac{(\lambda h)^i}{i!} - \lambda h - 1 \right) \\
&= e^{-\lambda h}(e^{\lambda h} - 1 - \lambda h) = 1 - e^{-\lambda h}(1 + \lambda h) \\
&= 1 - (1 - \lambda h + o(h))(1 + \lambda h) = 1 - (1 - \lambda^2 h^2 + o(h)) \\
&= \lambda^2 h^2 + o(h) = o(h).
\end{aligned}$$

Solution 2.2.6 $N(t) \sim \mathrm{P}(\lambda t)$.

Solution 2.2.7 First we show how to use conditioning.

$$\begin{aligned}
\mathbb{P}\{N_a(t)+N_s(t)=n\} &= \sum_{i=0}^{n} \mathbb{P}\{N_a(t)+N_s(t)=n|N_a(t)=i\}\,\mathbb{P}\{N_a(t)=i\} \\
&= \sum_{i=0}^{n} \mathbb{P}\{N_s(t)=n-i\}\,\mathbb{P}\{N_a(t)=i\} \\
&= \sum_{i=0}^{n} \frac{(\mu t)^{n-i}}{(n-i)!}\frac{(\lambda t)^i}{i!}e^{-(\mu+\lambda)t} \\
&= e^{-(\mu+\lambda)t}\sum_{i=0}^{n} \frac{(\mu t)^{n-i}}{(n-i)!}\frac{(\lambda t)^i}{i!} \\
&= e^{-(\mu+\lambda)t}\frac{1}{n!}\sum_{i=0}^{n}\binom{n}{i}(\mu t)^{n-i}(\lambda t)^i \\
&= \frac{((\mu+\lambda)t))^n}{n!}e^{-(\mu+\lambda)t}.
\end{aligned}$$

Now with moment generating functions.

$$\begin{aligned}
M_a(z) = \mathbb{E}\left[z^{N_a(t)}\right] &= \sum_{k=0}^{\infty} z^k\,\mathbb{P}\{N_a(t)=k\} \\
&= \sum_{k=0}^{\infty} z^k e^{-\lambda t}\frac{(\lambda t)^k}{k!} \\
&= e^{-\lambda t}\sum_{k=0}^{\infty}\frac{(z\lambda t)^k}{k!} \\
&= \exp(\lambda t(z-1)).
\end{aligned}$$

Use this, and the similar expression for $N_s(t)$ and independence to see that

$$M(z) = \mathbb{E}\left[z^{N(t)}\right] = \mathbb{E}\left[z^{N_a(t)}\right]\mathbb{E}\left[z^{N_s(t)}\right] = \exp((\lambda+\lambda)t(z-1)).$$

Finally, since the moment generating function uniquely characterizes the distribution, and since the above expression has the same form as $M_1(z)$ but with $\lambda+\mu$ replacing λ, we can conclude that $N(t) \sim P((\lambda+\mu)t)$.

Solution 2.2.8 With the above:

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

Solution 2.2.9 This means that, given that an event occurred, the event was an arrival, i.e., N_a was the first.

Solution 2.2.10 Suppose that N_1 is the thinned stream, and N the total stream. Then

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

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

Now consider $Y = \sum_{i=1}^{N} Z_i$. Suppose that $N = n$, so that n arrivals occurred. Then we throw n coins with success probability p. It follows that Y is indeed a thinned Poisson random variable. Model the coins as a generic Bernoulli distributed random variable Z. We first need

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

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

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

Then, using conditioning on N,

$$\begin{aligned}\mathbb{E}\left[e^{sY}\right] &= \mathbb{E}\left[\mathbb{E}\left[e^{s\sum_{i=1}^{n} Z_i}\,\middle|\, N=n\right]\right] = \mathbb{E}\left[\mathbb{E}\left[\left(1+p(e^s-1)\right)^n \,\middle|\, N=n\right]\right] \\ &= \sum_{n=0}^{\infty}\left(1+p(e^s-1)\right)^n e^{-\lambda}\frac{\lambda^n}{n!} = e^{-\lambda}\sum_{n=0}^{\infty}\frac{(1+p(e^s-1))^n\lambda^n}{n!} \\ &= e^{-\lambda}e^{\lambda(1+p(e^s-1))} = e^{\lambda p(e^s-1)}.\end{aligned}$$

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

Solution 2.2.11 I expect that this is easy for you by now. The exercise is just meant to help you recall this.

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

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

Similarly, using that $\mathbb{V}[N] = \mathbb{E}\left[N^2\right] - (\mathbb{E}[N])^2$,

$$\begin{aligned}\mathbb{E}\left[N^2\right] &= \sum_{n=0}^{\infty} n^2 e^{-\lambda}\frac{\lambda^n}{n!} \\ &= e^{-\lambda}\sum_{n=1}^{\infty} n\frac{\lambda^n}{(n-1)!} \\ &= e^{-\lambda}\sum_{n=0}^{\infty}(n+1)\frac{\lambda^{n+1}}{n!} \\ &= e^{-\lambda}\lambda\sum_{n=0}^{\infty} n\frac{\lambda^n}{n!} + e^{-\lambda}\lambda\sum_{n=0}^{\infty}\frac{\lambda^n}{n!} \\ &= \lambda^2+\lambda.\end{aligned}$$

The Poisson *process* $\{N(t)\}$ is a much more complicated object than the Poisson distributed random variable $N(t)$. The process contains it is an *uncountable set* of random variables, whereas $N(t)$ is just *one* random variable. However, for a fixed t the element $N(t)$ is a random variable that is Poisson distributed with parameter λt. Using the above, the answer of the question follows immediately.

With generating functions we get the result without too much effort. Suppose N is a Poisson

distributed random variable. Then

$$\begin{aligned}\phi(z) &= \mathbb{E}\left[z^N\right] \\ &= \sum_{k=0}^{\infty} z^k \mathbb{P}\{N=k\} \\ &= \sum_{k=0}^{\infty} z^k \frac{\lambda^k}{k!} e^{-\lambda} \\ &= e^{-\lambda} \sum_{k=0}^{\infty} \frac{(z\lambda^k}{k!} \\ &= e^{-\lambda} e^{z\lambda} \\ &= e^{(z-1)\lambda}.\end{aligned}$$

Then $\mathbb{E}[N] = \mathrm{d}\phi(z)/\mathrm{d}z$ at $z = 1$. Now, $\phi'(1) = \lambda$. Also $\mathbb{E}[N(N-1)] = \phi''(1) = \lambda^2$. From this we see that $\mathbb{E}\left[N^2\right] = \phi''(1) + \phi(1) = \lambda^2 + \lambda$.

The SCV is defined as $\mathbb{V}[N(t)]/(\mathbb{E}[N(t)])^2$. From the above it follows that this is $(\lambda t)/(\lambda t)^2 = 1/\lambda t$. Clearly, as t increases, $1/\lambda t$ decreases.

There is point of confusion for some students. The SCV of an exponentially distributed random variable is 1; the SCV of the related Poisson process $N_\lambda(t)$ is *not* identically 1 for all t.

Solution 2.3.1

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

Solution 2.4.1 The assumption is that the jobs arrive at the start of period k, before service in period k starts, rather than at the end of the period. Therefore the arrivals at period k can also be served during period k.

Solution 2.4.2
All jobs that arrive such that the queue become larger than K must be dropped. Thus, the accepted arrivals a'_n in week n are such that $a'_n = \min\{a_n, K - Q_{n-1}\}$, where a_n are all arrivals in period n. The rest of the recursions remain the same.

Solution 2.4.3 Let's assume that a fraction p of the jobs is lost each period. We can make p time-dependent if we like, but for the moment we don't. The amount produced in period k is d_k. Thus, pd_k is the amount lost, neglecting rounding errors for the moment. Thus, pd_k items have to be fed back to the system in the next period to be remade. Therefore the total amount of arrivals in period $k+1$ is $a'_{k+1} = a_{k+1} + pd_k$, i.e., the external arrivals plus the extra items. Now use the standard recursions but with the $\{a'_k\}$ rather than $\{a_k\}$.

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

Solution 2.4.4 Suppose again that a fraction p is faulty. Since these faulty items require less processing time than a new job, the service capacity c_k, i.e., the number of jobs that can be processed in period k, is a bit bigger; part of the capacity is spent at new jobs but another part is spent on the faulty jobs. Let's assume that the repair of a faulty requires half of the work of a new job, and that the faulty jobs are processed with priority over the new jobs. Assume queue A contains the faulty items, and queue B the new jobs. Then the recursions become:

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

Solution 2.4.5 First consider the dynamics of the queue. Since the capacity is chosen at the start of the period:

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

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

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

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

Solution 2.4.6 First we need to implement the N-policy. For this we need an extra variable to keep track of the state of the server. Let $I_k = 1$ if the machine is on in period k and $I_k = 0$ if it is off. Then $\{I_k\}$ must satisfy the relation

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

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

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

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

The machine cost for period k is βI_k, because only when the machine is on we have to pay β, and the queueing cost is hQ_k. To determine the total switching cost is harder as we need to determine how often the machine has been switched on up to time T. Observe that the machine is switched on in period k if $I_{k-1} = 0$ and $I_k = 1$. Thus, whenever $I_k - I_{k-1} = 1$ the machine is

switched on, when $I_k - I_{k-1} = 0$ the state of the machine remains the same, and if $I_k - I_{k-1} = -1$ the machine is switched off. In other words $\max\{I_k - I_{k-1}, 0\}$ captures what we need. The total cost up to time T becomes:

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

Solution 2.4.7 .

Solution 2.4.8 One model could be to let the server only switch on when the queue is larger than some threshold t, and when the server is on, it works at rate c per period. In that case, $c_k = c\,\mathbb{1}_{Q_{k-1}>t}$.

Solution 2.4.9 Let c_k^i be the capacity allocated to queue i in period k. The fair rule gives that

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

Then,

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

and likewise for the other queue.

Solution 2.4.10 If other choices for the division of the production capacity over the types is made (for instance strict priority to type A), it is necessary to split the queue in front of the mixing station into two separate queues Q_A and Q_B, one for each type. The rules below implement a strict priority rule for jobs type A.

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

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

Solution 2.4.11 First determine how much capacity queue B minimally needs in period k:

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

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

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

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

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

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

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

Solution 2.4.12 Let r_A be the reserved for queue A, and likewise for r_B. We assume of course that $c_k \geq r_A + r_B$, for all k.

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

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

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

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

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

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

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

Solution 2.4.13 Let a_k be the external arrivals at station A. Then:

$$\begin{aligned} d_k^A &= \min\{Q_{k-1}^A, c_k^A\}, \\ Q_k^A &= Q_{k-1}^A - d_k^A + a_k. \end{aligned} \tag{6.1}$$

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

$$\begin{aligned} a_k^B &= d_k^A, \\ d_k^B &= \min\{Q_{k-1}^B, c_k^B\}, \\ Q_k^B &= Q_{k-1}^B - d_k^B + a_k^B. \end{aligned} \tag{6.2}$$

Solution 2.4.14

$$\begin{aligned} d_k^A &= \min\{Q_{k-1}^A, c_k^A, M - Q_{k-1}^B\}, \\ Q_k^A &= Q_{k-1}^A - d_k^A + a_k, \\ a_k^B &= d_k^A, \\ d_k^B &= \min\{Q_{k-1}^B, c_k^B\}, \\ Q_k^B &= Q_{k-1}^B - d_k^B + a_k^B. \end{aligned} \tag{6.3}$$

This is a bit subtle: since there is room $M - Q_{k-1}^B$ at the intermediate buffer and $d_k^A \le M - Q_{k-1}^B$, we know that in the worst case, i.e., when $c_k^B = 0$, still $Q_k^B = Q_{k_1}^B + d_k^A$. Thus, we are sure that the queue length of the intermediate queue will not exceed M.

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

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

Solution 2.4.15 Realize that Stations A and B have their own arrivals.

$$\begin{aligned} d_k^A &= \min\{Q_{k-1}^A, c_k^A\}, \\ Q_k^A &= Q_{k-1}^A - d_k^A + a_k^A, \\ d_k^B &= \min\{Q_{k-1}^B, c_k^B\}, \\ Q_k^B &= Q_{k-1}^B - d_k^B + a_k^B, \\ a_k^C &= d_k^A + d_k^B, \\ d_k^C &= \min\{Q_{k-1}^C, c_k^C\}, \\ Q_k^C &= Q_{k-1}^C - d_k^C + a_k^C. \end{aligned} \tag{6.4}$$

Solution 2.4.16 The behavior of the queue length process is easy:

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

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

The only remaining problem is to make a model to 'distribute' the time between $l_{k,-}$ and $l_{k,+}$ over the two streams. One way is to give priority to a customers. Making this completely explicit, so that the recursion can be fed to the computer, requires still a bit of work. It is important to understand the trick we will discuss now because we will use it to model queueing systems with batching. Observe that the first job of the a stream only has to wait for $l_{k,-}$, the second job must wait $l_{k,-} + 1$, and so on. Thus, the average waiting is

$$\frac{l_{k,-} + (l_{k,-} + 1) + \ldots + (l_{k,-} + a_k^a - 1)}{a_k^a} = l_{k,-} + \frac{a_k^a - 1}{2}.$$

Similarly, the average waiting time for the b jobs must be

$$\frac{l_{k,-}+a_k^a+l_{k,-}+a_k^a+1+\ldots+l_{k,-}+a_k^a+a_k^b-1}{a_k^b}=l_{k,-}+a_k^a+\frac{a_k^b-1}{2}.$$

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

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

Solution 2.4.17

1. Realize that the recursions of Eq (2.6) applied to the queueing situation at the first machine provide us with the total number of departures d_k during period k. However, it does not tell us about the type of these departures. Thus, to compute the queue in front of station A, we need do know the number of departures of type A, rather than the total number of departures of the first station.

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

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

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

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

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

Solution 2.4.18 The decision variables are X_k, Y_K and S_k (note, it is not necessary to meet all demand: the production cost and profit may vary per period.) The objective is

$$\max\sum_{k=1}^{T}(r_kS_k-c_kX_k-d_kY_k-h_kI_k).$$

The constraints are

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

Solution 2.4.19

1. A job that arrives at time k cannot be served before period

$$l_-(k) := \max\left\{ l : \sum_{i=n}^{l} d_i < Q_{k-1} \right\}.$$

2. It will be served before period

$$l_+(k) := \min\left\{ l : \sum_{i=n+1}^{l} d_i \geq Q_k \right\}.$$

3. Thus, the waiting time of jobs arriving in period k must lie in $[l_-(k), l_+(k)]$.

Solution 2.4.20 Here is an example in python; please read the code to see how I approach the problem. The code is really easy, as it is nearly identical to the mathematical specification. You don't have to memorize the specific syntax of the code, of course, but it is (at least I find it) interesting to see how little code is actually necessary to set things up.

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

```
import numpy as np

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

labda = 20
mu = 21

a = np.random.poisson(labda, 10)
```

These are the number of arrivals in each period.

```
print(a)
```

```
[21 17 14 10 22 22 17 17 19 21]
```

The number of potential services

```
c = mu * np.ones_like(a)
print(c)
```

```
[21 21 21 21 21 21 21 21 21 21]
```

Now for the queueing recursions:

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

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

These are the departures for each period:

```
print(d)
```

```
[ 0 10 17 14 10 21 21 19 17 19]
```

The queue lengths

```
print(Q)
```

```
[10 17 14 10 22 23 19 17 19 21]
```

```
loss = (Q > 20)
print(loss)
print(loss.sum())
```

```
[False False False False  True  True False False False  True]
3
```

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

Now all statistics:

```
print(d.mean(), Q.mean(), Q.std(), (Q > 20).sum())
```

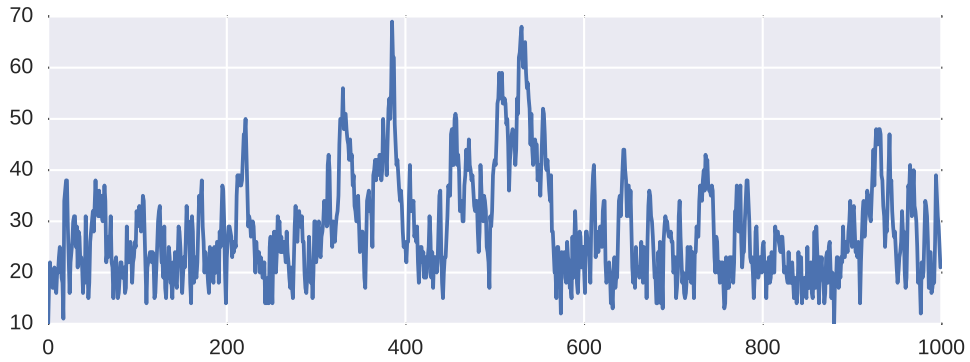
```
14.8 17.2 4.37721372565 3
```

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

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

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

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

print(d.mean(), Q.mean(), Q.std(), (Q > 40).mean()*100)

plt.figure(figsize=(6, 2))
plt.plot(Q)
# plt.show()
```

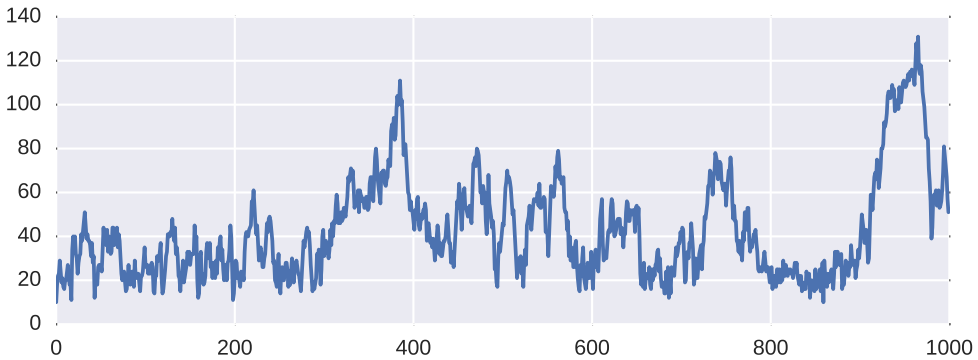
```
filename = "figures/md1_sim.pdf"
plt.savefig(filename)
plt.close()

20.178 28.42 10.1558653004 13.1
```



I multiply the mean of `(Q>20)` by 100 to get a percentage.

And finally an example with the same arrival pattern but with a Poisson distributed number of jobs served per day.

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

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

print(d.mean(), Q.mean(), Q.std(), (Q > 40).mean()*100)

plt.figure(figsize=(6, 2))
plt.plot(Q)
filename = "figures/mm1_sim.pdf"
plt.savefig(filename)
plt.close()

20.148 42.518 22.8410962084 44.0
```



You should compare the scales of these two graphs. The second queue behaves more wildly,

as expected.

You can try to implement the other cases below and analyze them in the same way. Hopefully you understand from the above that once you have the recursions to construct/simulate the queueing process, you are 'in business'. The rest is easy, just make plots, do some counting, and so on.

Solution 2.5.1 The intent of this exercise is to make you familiar with the notation.

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

Solution 2.5.2 First find the distribution of $U = S - X$ so that we can write $W_{Q,k} = [W_{Q,k-1} + U_k]^+$. Use independence of S and X:

$$\begin{aligned}
\mathbb{P}\{U=-2\} &= \mathbb{P}\{S-X=-2\} = \mathbb{P}\{S=1, X=3\} = \mathbb{P}\{S=1\}\,\mathbb{P}\{X=3\} = \frac{1}{4}\\
\mathbb{P}\{U=-1\} &= \mathbb{P}\{S=2\}\,\mathbb{P}\{X=3\} = \frac{1}{4}\\
\mathbb{P}\{U=0\} &= \mathbb{P}\{S=1\}\,\mathbb{P}\{X=1\} = \frac{1}{4}\\
\mathbb{P}\{U=1\} &= \mathbb{P}\{S=2\}\,\mathbb{P}\{X=1\} = \frac{1}{4}.
\end{aligned}$$

With this

$$\begin{aligned}
\mathbb{P}\{W_{Q,1}=1\} &= \mathbb{P}\{W_{Q,0}+U=1\} = \mathbb{P}\{3+U=1\} = \mathbb{P}\{U=-2\}\\
\mathbb{P}\{W_{Q,1}=2\} &= \mathbb{P}\{3+U=2\} = \mathbb{P}\{U=-1\}\\
\mathbb{P}\{W_{Q,1}=3\} &= \mathbb{P}\{3+U=3\} = \mathbb{P}\{U=0\}\\
\mathbb{P}\{W_{Q,1}=4\} &= \mathbb{P}\{3+U=4\} = \mathbb{P}\{U=1\}
\end{aligned}$$

And, then

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

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

```
>>> from lea import Lea
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
ImportError: No module named 'lea'
>>>
>>> W = 3
>>> S = Lea.fromVals(1,  2)
```

```
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>> X = Lea.fromVals(1,  3)
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>>
>>> # This is WQ1
>>> W = Lea.fastMax(W + S - X, 0)
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>> W
3
>>>
>>>
>>> # This is WQ2
>>> W = Lea.fastMax(W + S - X, 0)
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>> W
3
```

Solution 2.5.3

Let us write $f(x)$ for the density of $\mathbb{P}\{S-X\le x\}$, i.e., $\mathbb{P}\{S-X\le x\}=\int_{-\infty}^{x} f(y)\,\mathrm{d}y$. (A formal point, why does $\mathbb{P}\{S-X\le x\}$ actually have a density? You can skip this detail if you like, but it is interesting to think about.)

With conditioning,

$$f(x)=\frac{1}{10}\int_0^{10}\mathbb{P}\{S-X=x|X=u\}\,\mathrm{d}u=\frac{1}{10}\int_0^{10}\mathbb{P}\{S-u=x\}\,\mathrm{d}u=\frac{1}{10}\int_0^{10}\mathbb{P}\{S=x+u\}\,\mathrm{d}u.$$

(Observe that we just use the formulas we are used to from conditioning on sets with positive probability. Recall, $\mathbb{P}\{A|B\}=\mathbb{P}\{AB\}/\mathbb{P}\{B\}$ only when $\mathbb{P}\{B\}>0$, and here $\mathbb{P}\{X=x\}=0$ for all x, since x consists of a single point and X is uniformly distributed.) This works in this case, but you should be careful when using it.)

Next, interpret $\mathbb{P}\{S=y\}$ as the density of $\mathbb{P}\{S\le y\}$. Then,

$$\mathbb{P}\{S=y\}=\begin{cases}0, & \text{for } y\notin[0,7],\\ \frac{1}{7}, & \text{for } y\in[0,7].\end{cases}$$

Therefore, while using that in the above integral that x is fixed,

$$\mathbb{P}\{S=x+u\}=\frac{1}{7}1\{-x\le u\le 7-x\},$$

from which

$$\begin{aligned}f(x)&=\frac{1}{70}\int_0^{10}1\{-x\le u\le 7-x\}\,\mathrm{d}u\\&=\frac{1}{70}\int_{-\infty}^{\infty}1\{0\le u\le 10\}1\{-x\le u\le 7-x\}\,\mathrm{d}u\\&=\frac{1}{70}\int_{-\infty}^{\infty}1\{\max\{0,-x\}\le u\le\min\{10,7-x\}\,\mathrm{d}u.\end{aligned}$$

First we make some simple observations. If $x > 7$ then $\min\{10, 7-x\} < 0$. This is smaller then $\max\{0, -x\}$. Thus, $f(x) = 0$ for $x \geq 7$. Likewise, when $x \leq -10$, $f(x) = 0$. Also, when $x \in [-3, 0]$ the max and min overlap. Thus, all in all,

$$f(x) = \begin{cases} 0, & \text{if } x \leq -10, \\ \frac{x+10}{70} & \text{if } x \in [-10, -3], \\ \frac{1}{10} & \text{if } x \in [-3, 0], \\ \frac{7-x}{70} & \text{if } x \in [0, 7], \\ 0, & \text{if } x > 7, \end{cases}$$

To check my solution, I went to Wolframalpha. This is what what I typed:

```
\int_{-\infty}^\infty Boole[0<=u<=10] Boole[-x<=u<=7-x] du,
```

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

```
integral\_(-infinity)^infinity Boole[0<=u<=10] Boole[-x<=u<=7-x] du
```

For your convenience, I also include the following code from Wolframalpa

```
Integrate[Boole[Max[0, -x] <= u <= Min[10, 7 - x]], {u, -Infinity,
Infinity}]
```

Wolframalpha is a great site. Please check it out if you haven't done so up to now.

Solution 2.5.4 $A(t)$ is the number of arrivals during $[0, t]$. Suppose that $A(t) = n$. This nth job arrived at time A_n. Thus, $A_{A(t)}$ is the arrival time of the last job that arrived before or at time t. In a similar vein, A_n is the arrival time of the nth job. Thus, the number of arrivals up to time n, i.e., $A(A_n)$, must be n.

Solution 2.5.5 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 $\max\{k : A_k \leq t\} = 3$.

Solution 2.5.6

1. Because $D_s(t) = D(t)$. Once customers leave the server, their service is completed, and they leave the queueing system.
2. All customers that left the system must have left the queue. Thus, $D_Q(t) \geq D(t)$.
3. Jobs in the system are in queue or in service.
4. $L_Q(t) = A(t) - D_Q(t)$. $L_S(t) = D_Q(t) - D(t)$. This is in line with the fact that $L(t) = L_Q(t) + L_s(t) = A(t) - D(t)$.
5. In that 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.

Solution 2.5.7

1.

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

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 $A\bar{D} = \mathbb{1}_{A_k \le t < D_k}$

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

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

2. 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 $D(D_{k-1}) = k - 1$,

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

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

Solution 2.5.8 Let $L_k = L(A_k-)$, i.e., the number of jobs in the system as 'seen by' job k. It must be that $L_k = k - 1 - D(A_k)$. 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_k must hold. Applying this to job $k-1$ we get that $L_{k-1} = k - 2 - D(A_{k-1})$.

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

$$L_k = L_{k-1} + 1 - \sum_{i=k-1-L_{k-1}}^{k-1} 1\{D_i < A_k\}.$$

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

The reason to start at $k-1-L_{k-1}$ 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 $k-1-L_{k-1}, k-L_{k-1}, \ldots, k-1$, could have left the system between the arrival of job $k-1$ and job k.

Solution 2.5.9 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,

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

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

Solution 2.5.10

1. 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 $D_k = S'_k + S_k$.

2. I found this not easy, to say the least... The problem is that in a multi-server queueing systems, unlike for single-server queues, jobs can overtake each other: a small job that arrives after a very large job can still leave the system earlier. After trying for several hours, I obtained an inelegant method. A subsequent search on the web helped a lot. The solution below is a modification of N. Krivulin, 'Recursive equations based models of queueing systems'.

   The recursions for the two-server system are this:

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

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

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

3. Similar reasoning for the $G/G/m$ queue seems to lead to the following.

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

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

Solution 2.5.11 You can check the files on `github` in the `progs` directory.

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

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

Solution 2.5.12 This question came up in class. I don't think this can be constructed as a straightforward recursion, and a search on the web lead to nothing. In case you can find a recursion, please let me know.

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

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

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

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

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

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

Solution 2.5.13 Again, I don't know a good set of recursions, but an algorithm is straightforward. Rather then sorting the jobs in queue in ascending order of arrival time, just sort them in descending order of arrival time.

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

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

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

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

Solution 2.6.1 In the scheme

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

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

When we use the other scheme,

$$Q_k = [Q_{k-1} + a_k - c_k]^+,$$
$$d_k = Q_{k-1} + a_k - Q_k,$$

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

Solution 2.6.2 Note first that from the expression for Z_k, $a_k - c_k = Z_k - Z_{k-1}$. Using this in the recursion for Q_k, we get

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

thus,

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

From this, and using recursion, we see that

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

where we subsequently use that

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

Solution 2.6.3 With this we have a characterization of the queue length process as a function of time until it hits zero for the first time. What can we say about the distribution of $Q(t)$? With the above random walk,

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

We can write this a bit simpler by noting that

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

With this,

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

Solution 2.6.4 We use that merging two Poisson processes leads to another Poisson process and that a Poisson process can be split with Bernoulli random variables to form a thinned Poisson process. See one of the exercises of Section 2.2.

Consider the Poisson process $N_{\lambda+\mu}(t)$. This counting process has, for fixed t, the same distribution as $N_\lambda(t)+N_\mu(t)$. Let $\{A_k\}$ be the epochs at which the arrivals of $N_{\lambda+\mu}(t)$ occur. At A_k, throw a coin that lands heads with probability $\lambda/(\lambda+\mu)$ and tails with probability $\mu/(\lambda+\mu)$. Set $a_k = 1$ and $c_k = 0$ if the coin lands heads, and set $a_k = 0$ and $c_k = 1$ if it turns tails. The processes

$$\begin{aligned}
N_\lambda(t) &= \sum_{k=1}^{\infty} a_k \mathbb{1}_{A_k\le t},\\
N_\mu(t) &= \sum_{k=1}^{\infty} c_k \mathbb{1}_{A_k\le t},
\end{aligned}$$

are our original Poisson processes. Observe that with this procedure we have established a relation between the sequences $\{a_k\}$ and $\{c_k\}$ and the Poisson processes N_λ and N_μ.

The process $Z_k = Z_{k-1} + a_k - c_k$ is a random walk in discrete time and its reflection results in the discrete-time $M/M/1$ queue. The process $Z(t) = Z(0) + N_\lambda(t) - N_\mu(t)$, with N_λ and N_μ constructed based on the sequences $\{a_k\}$ and $\{c_k\}$, is the continuous-time random walk underlying the $M/M/1$ queue.

Solution 2.6.5 Here is an example with python. In R it must be equaly simple. I compute the following

$$\max_x \{|\,\mathbb{P}\{W_{Q,k} \le x\} - \mathbb{P}\{W_{Q,k-1} \le x\}\,|\},$$

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

```
>>> from lea import Lea
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
ImportError: No module named 'lea'
>>>
>>> S = Lea.fromVals(1,  2)
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>> X = Lea.fromVals(1,  3)
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>> U = S-X
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'S' is not defined
>>>
>>> W = Lea.fromVals(3)
Traceback (most recent call last):
  File "< chunk 1 named None >", line 1, in <module>
NameError: name 'Lea' is not defined
>>> for k in range(1, 10):
...     W_new = Lea.fastMax(W + U, 0)
...     m = max(abs(W_new.cdf(x)-W.cdf(x)) for x in
range(W_new.support()[-1]))
...     print(k, m)
...     W = W_new
...
Traceback (most recent call last):
  File "< chunk 1 named None >", line 2, in <module>
NameError: name 'Lea' is not defined
```

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

Solution 2.6.6 You can study `waiting_time_simulation.py` at github if you like, but skipping it is OK. Trying to make the graph in your favorite programming language is fun.

Solution 2.7.1

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

Solution 2.7.2 That the average time customers spend in service is smaller than the average time between the arrival of two subsequent jobs.

Solution 2.7.3 $0 > \mathbb{E}[U_k] = \mathbb{E}[S_{k-1} - X_k] = \mathbb{E}[S_{k-1}] - \mathbb{E}[X_k] = \mathbb{E}[S] - \mathbb{E}[X]$, where we use the fact that the $\{S_k\}$ and $\{X_k\}$ are i.i.d. sequences. Hence, $\mathbb{E}[S] - \mathbb{E}[X] = (\mathbb{E}[S]/\mathbb{E}[X] - 1)\mathbb{E}[X] = (\lambda/\mu - 1)\mathbb{E}[X] = (\lambda/\mu - 1)\mathbb{E}[X]$.

Solution 2.7.4 Let $T_1 > A_1$ be the first time after the arrival of job 1 that arrives at an empty system. (Observe that job 1 also arrives at an empty system.) Suppose, for ease of writing, that this job is the $n+1$th job, so that up to time T_1 the number of arrivals is n. Since the first job arrived at time $A_1 = X_1$, the first n jobs arrived during $[X_1, T_1)$. The total amount of service that arrived during this period is $\sum_{i=1}^{n} S_i$. Thus the fraction of time that the server has been busy during $[X_1, T_1)$ is

$$\frac{\sum_{i=1}^{n} S_i}{T_1 - X_1} = \frac{\sum_{i=1}^{n} S_i}{\sum_{i=1}^{n+1} X_i - X_1} = \frac{\sum_{i=1}^{n} S_i}{\sum_{i=2}^{n+1} X_i}$$

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

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

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

Solution 2.7.5 Since the fraction of idle time is 1 minus the fraction of busy time, it follows that $1 - \mathbb{E}[S]/\mathbb{E}[X] = (\mathbb{E}[X] - \mathbb{E}[S])/\mathbb{E}[X]$ is the idle time fraction.

Solution 2.7.6 Consider a busy cycle, that is, a cycle that starts with the first job that sees an empty system at upon arrival up to the time another job sees an empty system upon arrival. In such one cycle, the server is busy for an expected duration $\mathbb{E}[B]$. The total expected length of the cycle is $\mathbb{E}[B] + \mathbb{E}[I]$, since after the last job of the cycle left, the expected time until the next job is $\mathbb{E}[I]$.

Since ρ is utilization of the server,

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

With a bit of algebra the result follows.

Solution 2.7.7 The criterion is $\rho < c$. 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 both systems is the same, i.e., $c\mu$, where μ is the rate of one server. For the system with the fast server $\lambda/c\mu < 1$, from which follows that $\lambda/\mu < c$. 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. Again, applying the condition that $(\lambda/c)/\mu < 1$ per server leads to the same conditions.

Later we define the load ρ for the utilization of a multi-server queue as $\lambda/c\mu$.

Solution 2.7.8 Realize that the machine works in cycles. A cycle starts with processing k_A jobs of type A, then does a setup, and processes k_B jobs of type B, and then a new cycle starts again. The time it takes to complete one such cycle is $T = k_A t_A + S + k_B t_B$. The number of jobs of type A processed during one such cycle is, of course, k_A. Observe next that the average number of jobs that arrive during one cycle is $\lambda_A T$. We of course want that $\lambda_A T < k_A$, i.e., less jobs of type A arrive on average per cycle than what we can process.

Solution 2.7.9 Observing that $A_{A(t)}$ is the arrival time of the last job before time t and that $A_{A(t)+1}$ is the arrival time of the first job after time t:

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

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

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

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

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

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

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

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

Solution 2.7.10 See my website.

Solution 2.8.1 Take $X_k = 10$ and $S_k = 10 - \epsilon$ for some tiny ϵ. Then $L(t) = 1$ nearly all of the time. In fact, $\mathbb{E}[L] = 1 - \epsilon/10$. However, $L_k = 0$ for all k.

Solution 2.8.2

1. $\lambda = 6$ per hour, and $\mu = 60/11$ per hour. Note that $\mu < \lambda$. $A_0 = 0$, $A_1 = 10$, $A_2 = 20$, etc., hence $A_k = 10k$. $W_{Q,0} = 0$, $W_{Q,1} = \max\{0 + 0 - 10, 0\} = 0$. $W_{Q,2} = \max\{0 + 11 - 10, 0\} = 1$. $W_{Q,3} = \max\{1 + 11 - 10, 0\} = 2$. Hence, $W_{Q,k} = k - 1$ for $k \ge 1$. Thus, $W_k = k - 1 + 11 = k + 10$ for $k \ge 1$, and $D_k = 10k + k + 10 = 11k + 10$. Note that W_k increases linearly as a function

of k. You can infer that, for the queue length to remain bounded, it is necessary that the service rate $\mu > \lambda$.

2. Trivial.

Solution 2.8.3 Since we deal with a system in discrete time L_k is the queue length at the end of period k. Thus, $\sum_{k=1}^{n} \mathbb{1}_{L_k > m}$ counts the number of *periods* that the queue is larger than m. This is of course not the same as the number of *jobs* that see a queue larger than m; only when $a_k > 0$ the customers in a batch would see a queue $L_k > m$. Thus,

$$\sum_{k=1}^{n} \mathbb{1}_{L_k > m} \mathbb{1}_{a_k > 0},$$

counts the number of batches.

Next, by assumption, a_k customers arrive during period k. The first of these customers sees a queue length of $L_{k-1} - d_k$, the second $L_{k-1} - d_k + 1$, and so on until the last customer who sees a queue length of $L_{k-1} - d_k + a_k - 1 = L_k - 1$. Thus, of all jobs the last sees the largest queue. Hence, if $L_k \leq m$, all customers of the batch see a queue less than m. If, however, $L_k > m$, then $L_k - m$ customers saw m or more jobs in the system.. Therefore, the fraction of arrivals that see a queue with m or more jobs is equal to

$$\frac{1}{A(n)} \sum_{k=1}^{n} (L_k - m) \mathbb{1}_{L_k > m}.$$

We could also define this a bit differently. Suppose that we don't want to distinguish between jobs in a batch, but simply want to say that if one job sees a long queue, all see a long queue. In that case,

$$\frac{1}{A(n)} \sum_{k=1}^{n} a_k \mathbb{1}_{L_k > m}.$$

Thus, when jobs arrive in batches, the definition of loss fraction requires some care; not all definitions need to measure the same.

Solution 2.9.1 $L(t)$ is the number of customers in the system at time t. As such the function $t \to L(t)$ is *right-continuous*. The definition of $L(A_k -) = \lim_{t \uparrow A_k} L(t)$, i.e., the limit from the left. The customer therefore 'sees' $L(A_k -)$ just before he arrives.

Solution 2.9.2 $D(n-1, t)$ counts the departures that leave $n-1$ behind. Thus, just before the customer leaves, the system contains n customers.

Solution 2.9.3 $A(t)$ counts all customers that arrive up to time t, i.e., during $[0, t]$. Note that this *includes* time t. $A(n, t)$ counts the jobs that see n jobs in the system just before they arrive.

Solution 2.9.4 Observe that $\mathbb{1}_{A_k \leq t} \mathbb{1}_{L(A_k -) = n} = \mathbb{1}_{A_k \leq t} \mathbb{1}_{L(A_k -) = n} \leq \mathbb{1}_{A_k \leq t}$; the last inequality follows from the fact that $\mathbb{1}_{L(A_k -) = n} \leq 1$. In fact, most of the time, for a particular n, $\mathbb{1}_{L(A_k -) = n} = 0$. Therefore

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

For any 'normal' queueing system, $A(t) > A(t, n)$, because the queue length fluctuates.

Solution 2.9.5 If $\lambda > \gamma$, then $L(t) \to \infty$. But then there must be last time, s say, that $L(s) = n+1$, and $L(t) > n+1$ for all $t > s$. Hence, after time s no job will see the system with n jobs. Thus $A(t,n) = A(s,n)$ for all $t > s$. This is a finite number, while $t \to \infty$, so that $A(n,t)/t \to 0$.

Solution 2.9.6 When $\gamma < \lambda$, λ can still be finite, but the server is simply to slow. In that case both limits can exists, the first limit is positive but the second is 0.

Solution 2.9.7 First, it is the $D/D/1$ queue, since there is one server and the interarrival times and service times are constant, i.e., Deterministic.

Next, to get $Y(n,t)$, observe that the system never contains more than 1 job. Hence, $Y(n,t) = 0$ for all $n \geq 2$. Then, make a plot of $L(s)$ for $n = 1$ as a function of time. function of time. 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 $t \in [1,2)$, $L(s) = 0$, so that $Y(1,t) = 1$ for $s \in [1,2)$, and so on. Thus, all in all,

$$Y(1,t) = \begin{cases} t & \leq t \in [0,1), \\ 1 & \leq t \in [1,2), \\ 1+(t-2) & \leq t \in [2,3), \\ 2 & \leq t \in [3,4), \\ 2+(t-4) & \leq t \in [4,5), \end{cases}$$

and so on. Since $Y(n,t) = 0$ for all $n \geq 2$, $L(s) = 1$ or $L(s) = 0$ for all s, therefore, $Y(0,t) = 1 - Y(1,t)$.

Solution 2.10.1

$$\begin{aligned}
\mathbb{E}[L] &= \sum_{n=0}^{\infty} n\pi(n) = \sum_{n=0}^{\infty}\sum_{i=1}^{n} 1\{i \leq n\}\pi(n) \\
&= \sum_{i=1}^{\infty}\sum_{n=0}^{\infty} 1\{i \leq n\}\pi(n) = \sum_{i=1}^{\infty}\sum_{n=i}^{\infty} \pi(n) \\
&= \sum_{i=1}^{\infty}\sum_{n=i}^{\infty} (1-\rho)\rho^n = \sum_{i=1}^{\infty}(1-\rho)\rho^i \sum_{n=0}^{\infty} \rho^n \\
&= \sum_{i=1}^{\infty}(1-\rho)\rho^i \frac{1}{1-\rho} = \sum_{i=1}^{\infty}\rho^i = \rho\sum_{i=0}^{\infty}\rho^i = \frac{\rho}{1-\rho}.
\end{aligned}$$

Solution 2.10.2

$$\begin{aligned}
\mathbb{P}\{L \geq n\} &= \sum_{k=n}^{\infty} p(n) = \sum_{k=n}^{\infty} p(0)\rho^n = (1-\rho)\sum_{k=n}^{\infty}\rho^k \\
&= (1-\rho)\rho^n \sum_{k=0}^{\infty}\rho^k = \rho^n \sum_{k=0}^{\infty}(1-\rho)\rho^k = \rho^n \sum_{k=0}^{\infty} p(k) = \rho^n.
\end{aligned}$$

Solution 2.10.3

Solution 2.11.1

1.

$$\lambda p(0) = \mu p(1)$$
$$\lambda p(1) = 2\mu p(2)$$
$$\lambda p(2) = 2\mu p(3)$$

2. From the above, $p(1) = \rho p(0)$, with $\rho = \lambda/\mu$, $p(2) = (\rho/2)p(1) = (\rho^2/2)p(0)$, and $p(3) = (\rho/2)p(2) = (\rho^3/4)p(0)$. Finally, normalize to arrive at $p(0) = (1 + \rho + \rho^2/2 + \rho^3/4)^{-1}$.

3. Compare Figure 2.11.

Solution 2.11.2 $\lambda(n) \equiv \lambda$ for all $n < K + k$. When $n = K + k$, $\lambda(K) = 0$, since then the system is full, and all arriving jobs will be dropped. When $n \le k$, $\mu(n) = n\mu$ since only n servers are active/occupied. When $n > k$, $\mu(n) = k\mu$. Thus,

$$p(n) = \frac{\lambda}{n\mu} p(n-1) = \frac{(\lambda/\mu)^n}{n!} p(0) = \frac{(k\rho)^n}{n!} p(0), \, n \le k,$$
$$p(n) = \frac{\lambda}{k\mu} p(n-1) = \frac{k^k \rho^n}{k!} p(0), \, k < n \le K + k,$$

where $\rho = \lambda/k\mu$.

The normalization is trivial, numerically at least.

Solution 2.11.3 Note that

$$1 = \sum_{i=0}^{K} p(i) = p(0) \sum_{i=0}^{K} \rho^i = p(0) \frac{1 - \rho^{K+1}}{1 - \rho}.$$

Thus, $G = 1/p(0)$ and the result follows.

From Eq. (2.41), $G \to (1-\rho)^{-1}$ as $\rho \to K$. 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.

Solution 2.11.4

$$\begin{aligned}
p(n+1) &= \frac{\lambda}{\mu(n+1)} p(n) \\
&= \frac{\lambda}{\min\{c, n+1\}\mu} p(n) \\
&= \frac{1}{\min\{c, n+1\}} \frac{\lambda}{\mu} p(n) \\
&= \frac{1}{\min\{c, n+1\}} (c\rho) p(n) \\
&= \frac{1}{\min\{c, n+1\} \min\{c, n\}} (c\rho)^2 p(n-1) \\
&= \frac{1}{\Pi_{k=1}^{n+1} \min\{c, k\}} \rho^{n+1} p(0).
\end{aligned}$$

Write $p(0) = 1/G$.

For G,

$$\begin{aligned} G &= \sum_{i=0}^{c-1} \frac{(c\rho)^n}{n!} + sum_{i=c}^{\infty} \frac{(c\rho)^c}{c!} \rho^{n-c} \\ &= \sum_{i=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!} sum_{i=c}^{\infty} \rho^n \\ &= \sum_{i=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}. \end{aligned}$$

Next,

$$\begin{aligned} \mathbb{E}\left[L_Q\right] &= \sum_{n=c}^{\infty} (n-c) p(n) \\ &= \frac{c^c \rho^c}{Gc!} \sum_{n=c}^{\infty} (n-c) \rho^{n-c} \\ &= \frac{c^c \rho^c}{Gc!} \sum_{n=0}^{\infty} n \rho^n \end{aligned}$$

Then,

$$\begin{aligned} \sum_{n=0}^{\infty} n\rho^n &= \sum_{n=0}^{\infty} \sum_{i=1}^{\infty} \mathbb{1}_{i \le n} \rho^n \\ &= \sum_{i=1}^{\infty} \sum_{n=0}^{\infty} \mathbb{1}_{i \le n} \rho^n \\ &= \sum_{i=1}^{\infty} \sum_{n=i}^{\infty} \rho^n \\ &= \sum_{i=1}^{\infty} \rho^i \sum_{n=0}^{\infty} \rho^n \\ &= \frac{1}{1-\rho} \sum_{i=1}^{\infty} \rho^i \\ &= \frac{\rho}{1-\rho} \sum_{i=0}^{\infty} \rho^i \\ &= \frac{\rho}{(1-\rho)^2}. \end{aligned}$$

Observe again that using indicators and Fubinaci's theorem (interchanging summations/integrals) makes the above computation painless.

We next show that the expected number of jobs in the system is given by

$$\mathbb{E}[L_S] = \sum_{n=0}^{c-1} n p(n) + \sum_{n=c}^{\infty} c p(n).$$

This expression is not the easiest to start with. With a slight modification the entire derivation becomes easier. I also premultiply by the normalization constant G to get rid of it on the right

hand side.

$$\begin{aligned}
G\mathbb{E}[L_S] &= G\sum_{n=0}^{c} np(n) + \sum_{n=c+1}^{\infty} cp(n) \\
&= \sum_{n=1}^{c} n\frac{(c\rho)^n}{n!} + \sum_{n=c+1}^{\infty} c\frac{c^c\rho^n}{c!} \\
&= \sum_{n=1}^{c} \frac{(c\rho)^n}{(n-1)!} + \frac{c^{c+1}}{c!}\sum_{n=c+1}^{\infty} \rho^n \\
&= \sum_{n=0}^{c-1} \frac{(c\rho)^{n+1}}{n!} + \frac{(c\rho)^{c+1}}{c!}\sum_{n=0}^{\infty} \rho^n \\
&= c\rho\left(\sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)}\right).
\end{aligned}$$

Observe that the right hand side is precisely equal to ρcG.

Solution 2.11.5

1. The second term in (2.42c) is $(c\rho)^c/c! = (\lambda/\mu)^c/c!$. It is well known that $x^c/c! \to 0$ as $c \to \infty$.

2. No matter how many jobs are in service, there is always another free server available when a new job arrives. Thus, jobs never have to wait in queue, and only spend time in service. Since $\mathbb{E}[S] < \infty$ by assumption, jobs spend a finite time (with probability one) at a server.

3. Write $\rho = \lambda/\mu$. Then $p(n) = e^{-\rho}\rho^n/n!$. Thus, $p(n)$ is the Poisson distribution with parameter ρ, and L is a random variable with this Poisson distribution. Thus, $\mathbb{E}[L] = \rho$.

Solution 2.11.6

1. Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = \mu$, and solve Eq. (2.34) and (2.36). Thus:

$$\begin{aligned}
p(n+1) &= \frac{(N-n)\lambda}{\mu}p(n) \\
&= \rho(N-n)p(n) \\
&= \rho^2(N-n)(N-(n-1))p(n-1) \\
&= \rho^3(N-n)(N-(n-1))(N-(n-2))p(n-2) \\
&= \rho^{n+1}(N-n)(N-(n-1))\cdots(N-(0))p(0) \\
&= \rho^{n+1}\frac{N!}{(N-(n+1))!}p(0).
\end{aligned}$$

   Next, we need to normalize this. Observe that $p(N+1) = P(N+2) = \ldots = 0$ since there are just N customers, so that the system can never contain more than N customers. Thus, we want $p(0)$ to be such that

$$1 = \sum_{n=0}^{N} p(n) = p(0)\sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!}$$

We see from this that $p(0)$ times some constant must be 1. Hence, dividing by this constant, we get

$$p(0) = \left(\sum_{n=0}^{N} \rho^n \frac{N!}{(N-n)!} \right)^{-1}.$$

I asked WolframAlpha to simplify this, but the answer I got was not particularly revealing.

2. 

3. Take $\lambda(n) = (N-n)\lambda$ and $\mu(n) = n\mu$. Then

$$\begin{aligned} p(n+1) &= \frac{\lambda(n)}{\mu(n+1)} p(n) \\ &= \frac{(N-n)\lambda}{(n+1)\mu} p(n) \\ &= \frac{(N-n)(N-(n-1))}{(n+1)n} \frac{\lambda^2}{\mu^2} p(n-1) \\ &= \frac{N!}{(N-(n+1))!} \frac{1}{(n+1)!} \rho^{n+1} p(0) \\ &= \binom{N}{n+1} \rho^{n+1} p(0). \end{aligned}$$

Hence, after normalization, i.e., requiring that $p(0)$ is such that $\sum_{n=0}^{N} p(n) = 1$, so that $p(0) = \left(\sum_{k=0}^{N} \rho^k \binom{N}{k} \right)^{-1}$, the final result becomes

$$p(n) = \frac{\rho^n \binom{N}{n}}{\sum_{k=0}^{N} \rho^k \binom{N}{k}}.$$

Solution 2.11.7 Finite buffer size. Formally the number of customers that fit into a shop is necessarily finite. This answer, however, is not intended. Typically, the number of customers in a restaurant is limited. Example 2: Sometimes call centers reject callers when the system is too busy.

A finite calling population occurs for instance at a factory with a number of machines. When a machine breaks down, it becomes a (repair) job at the repair department. Thus, a break down forms an arrival at the repair shop. The mechanics at the repair department form a set of parallel servers. Typically, the number of machines quite small, 10 or so, and when a machine is 'down', i.e., broken, it cannot break again. Hence, when 2, say, machines are in repair, the number of 'customers' that can arrive to the queueing system is only 8.

Solution 2.11.8 In a queueing system with balking, customers may decide to balk at a level b. Thus, whether only b customers are admitted to the system (i.e., blocked), or balk at level b, the effect is the same: the number of people in the system remains at or below b. However, a fraction of customer may already balk at lower levels, like in the example above, so that the arrival stream is 'thinned' due to balking customers. In that respect, a queueing system with balking behaves differently.

Solution 2.12.1 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 $t \to \infty$.

Solution 2.12.2

1. $A(t) = \lfloor t \rfloor$ provided the unit of t is in hours. $A(0,t) = A(t)$ and $A(n,t) = 0$ for $n > 0$. Thus,

$$
\begin{aligned}
A(0,t) &\approx t \\
A(1,t) &= 0 \\
Y(0,t) &= \frac{1}{60}\lfloor t \rfloor + (t - \lfloor t \rfloor)1\{t - \lfloor t \rfloor \in [59/60, 1)\} \approx \frac{1}{60}t, \\
Y(1,t) &= \frac{59}{60}\lfloor t \rfloor + (t - \lfloor t \rfloor)1\{t - \lfloor t \rfloor \in [0, 59/60)\} \approx \frac{59}{60}t. \\
\lambda(0) &= \lim_t \frac{A(0,t)}{Y(0,t)} = \lim_t \frac{t}{t/60} = 60, \\
\lambda(1) &= \lim_t \frac{A(1,t)}{Y(1,t)} = \lim_t \frac{0}{59/60t} = 0 \\
\lambda(n) &= 0, \quad n \geq 1. \\
\lambda &= \lim_t \frac{A(t)}{t} = 1 \\
p(0) &= \frac{1}{60}, \\
p(1) &= \frac{59}{60}, \\
\rho &= \frac{59}{60}, \\
\pi(0) &= 1.
\end{aligned}
$$

   There is no queue so $\mathbb{E}\left[L_Q\right] = 0$, but $\mathbb{E}[L] = \rho$. The queue length as observed by customers is equal to 0, because jobs only arrive when the server is free.

2. We have to check that $\lambda(n)p(n) = \lambda\pi(n)$. First take $n = 0$. By the above: $\lambda(0)p(0) = 60 \cdot 1/60 = 1$, $\lambda\pi(0) = 1 \cdot 1 = 1$. For $n = 1$: $\lambda(1)p(1) = 0$. Since $\pi(1) = 0$ the result is again OK for $n = 1$. And so on .

3. Again $\rho = 59/60$.

   Observe that the average queue length is very different from the previous example, even though $\rho = 59/60$ in both cases. Thus, knowledge of the load is not sufficient to make a statement about the average queue length.

   To compute the average queue length, as perceived by the customers, note that the first customer sees no queue, the second sees one customer in front, and so on. Thus, the average queue length as seen by customers is

$$\frac{0 + 1 + 2 + \cdots + 58}{59} = \frac{58 \cdot 59}{2 \cdot 59} = 29.$$

   The time average queue length is $\mathbb{E}[L] = 59/60 + 58/60 + \cdots 1/60 + 0/60 = 59/2$.

These two examples show that the way customers arrive have an enormous impact on the queue length. Both queueing systems have the same utilization, i.e. 59/60, but the average queue lengths are not nearly the same, i.e. 0 versus 29.

Solution 2.12.3 We have shown for one-step transition processes that $\lambda\pi(n) = \gamma\delta(n)$. Thus, $\pi(n) = \gamma/\lambda\delta(n)$. Since $\lambda \geq \gamma$, we have that $\pi(n) \geq \delta(n)$.

Solution 2.12.4 The $M/M/1$-queue can be constructed as a reflection of the random walk $Z(t) = Z(0) + N_\lambda(t) - N_\mu(t)$. Clearly, downcrossings can only occur when N_μ fires. The rate at which the transitions of N_μ occur is constant, and, in particular, independent of the history of Z.

More specifically, for the interested, define $\sigma\{X(t) : t \in I\}$ as the σ-algebra generated by the stochastic procsses $\{X(t), t \in I\}$ on the index set I. Then, by construction of $\{N_\lambda(t)\}$ and $\{Z(t)\}$, we have that $\sigma\{Z(s) : s \in [0,t]\}$ and $\sigma\{N_\lambda(u) : u > t\}$ are independent.

Solution 2.12.5 The important condition is that transitions occur as single steps. In other words, the relation is true for processes with *one-step* transitions, i.e, when $|A(n,t) - D(n,t)| \leq 1$. In that case,

$$\begin{aligned}
\frac{A(n,t)}{t} &= \frac{A(n,t)}{A(t)}\frac{A(t)}{t} \to \pi(n)\lambda \\
\frac{A(n,t)}{t} &= \frac{A(n,t)}{Y(n,t)}\frac{Y(n,t)}{t} \to \lambda(n)p(n) \\
\frac{D(n,t)}{t} &= \frac{D(n,t)}{Y(n+1,t)}\frac{Y(n+1,t)}{t} \to \mu(n+1)p(n+1) \\
\frac{D(n,t)}{t} &= \frac{D(n,t)}{D(t)}\frac{D(t)}{t} \to \delta(n)\mu
\end{aligned}$$

Solution 2.13.1

1. This was my initial answer: '$\mathbb{E}[W] = \lambda\mathbb{E}[L] = \lambda 10$, and *not* 10μ.' Interestingly, it is wrong... I wrote Little's law in the wrong way... So, be aware! It's all too easy to make mistakes with Little's law.

   This is correct: $\mathbb{E}[W] = \mathbb{E}[L]/\lambda = 10/\lambda$, and *not* 10μ.

2. The time you spend in the system is the expected remaining service time $\mathbb{E}[S]_r$, i.e., the time to serve the customer in service at the moment of arrival, plus $10/\mu_1$, i.e., the time to clear the queue of 10 customers.

3. In the second question, it is *given* that the queue length is 10 at the moment of arrival. The queue length as 'seen' by a given customer upon arrival need not be the same as the time-average queue length.

Solution 2.13.2 The area enclosed between the graphs of $A(t)$ and $D(t)$ until T can be 'chopped up' in two ways: in the horizontal direction and the vertical direction. (Please make the drawing as you go along...) A horizontal line between $A(t)$ and $D(t)$ corresponds to the waiting time of a job, while a vertical line corresponds to the number of jobs in the system at time t. Now adding all horizontal lines (by integrating along the y-axis) makes up the total amount of

waiting done by all the jobs until time T, while adding the vertical lines (by integrating along the x-axis) is equal to the summation of all jobs in the system. Since the area is the same no matter whether you sum it in the horizontal or vertical direction:

$$\sum_{k=1}^{A(T)} W_k = \text{eclosed area} = \int_0^T (A(t) - D(t))\,dt.$$

Dividing both sides by $A(T)$ gives

$$\frac{1}{A(T)} \sum_{k=1}^{A(T)} W_k = \frac{1}{A(T)} \int_0^T (A(t) - D(t))\,dt.$$

Finally, observe that this equality holds between any two times T_i, T_{i+1}, where times $\{T_i\}$ are such that $A(T_i) = D(T_i)$. Then, as $T_i \to \infty$ which we assumed from the on-set, $\frac{1}{A(T_i)} \sum_{k=1}^{A(T_i)} W_k \to \mathbb{E}[W]$, and

$$\frac{T_i}{A(T_i)} \frac{1}{T_i} \int_0^{T_i} (A(t) - D(t))\,dt \to \lambda^{-1} \mathbb{E}[L].$$

Hence, Little's law follows.

Solution 2.13.3

1. Observe that the average time at the server, i.e., the waiting time in service, is $\mathbb{E}[S]$.
2. Jobs arrive at rate λ.
3. So, the time-average number of jobs at the server $\mathbb{E}[L]_S = \lambda \mathbb{E}[S]$.
4. Applied to a single server, $\mathbb{E}[L]_S$ is the long run fraction of time a job is at the server.
5. Thus, the long-run fraction of time the server is occupied is $\mathbb{E}[L]_S = \lambda \mathbb{E}[S]$. The latter is also defined as ρ. Hence, $\rho = \mathbb{E}[L]_S$.

Solution 2.13.4

1. - $A(t)/t \to \lambda$ as $t \to \infty$, i.e., $A(t)/t$ has a limit as t converges to ∞.
   - There exists a sequence of points $T_k, k = 0, 1, 2, \ldots$ in time such that the server is idle.
   - Either of the limits $\sum_k^n W_k/n = \sum_k^n S_k/n$ or $t^{-1} \int_0^t L(s)\,\mathrm{d}s$ exists, in which case the other exists.
2. If $N(t) = 3t^2$, then clearly $N(t)/t = 3t$. This does not converge to a limit. For the mathematically interested, we seek a function for which its Cesaro limit does not exist. Another example, let the arrival rate $\lambda(t)$ be given as follows:

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

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

Solution 2.14.1 This is, admittedly, not simple, so let us work in stages. We first refine the definition of the number of arrivals $A(t)$ up to time t to the number of arrivals $A(+,t)$ that see a job in service. As the server is only busy at time t when $L(t) > 0$, we define

$$A(+,t) = \sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t, L(A_k)>0}.$$

Next, define $\tilde{D}_k = \min\{D_i : D_i \ge A_k\}$ as the first departure after the arrival time A_k of job k. We need to consider two cases. If $\tilde{D}_k = D_k$ then, obviously, the first departure after A_k coincides with the departure time of job k. This is only possible if $L(A_k) = 0$. But then, it must be that the remaining service time of the job in service at time A_k is 0, as there is no job in service. Otherwise, if $L(A_k) > 0$ it must be that $\tilde{D}_k < D_k$, and, as a consequence, the remaining service time of the job in service at time A_k is equal to $\tilde{D}_k - A_k$. All in all, the total amount of remaining service times added up for all arrivals up to time t can be written as

$$S_r(t) = \sum_{k=1}^{\infty} (\tilde{D}_k - A_k)\, \mathbb{1}_{A_k \le t, L(A_k)>0}.$$

The remaining service time averaged over all arrival times up to time t is therefore $S_r(t)/A(t)$. We can now rewrite this to

$$\frac{S_r(t)}{A(t)} = \frac{A(+,t)}{A(t)} \frac{S_r(t)}{A(+,t)},$$

and interpret these fractions. First, consider

$$\frac{A(+,t)}{A(t)} = \frac{\sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t, L(A_k)>0}}{\sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t}}.$$

This is the number of jobs up to time t that see at least one job in the system divided by the total number of arrivals up to time t. Thus, as $t \to \infty$,

$$\frac{A(+,t)}{A(t)} \to \sum_{n=1}^{\infty} \pi(n) = 1 - \pi(0).$$

With PASTA we can conclude that $1 - \pi(0) = 1 - p(0) = \rho$. Second,

$$\frac{S_r(t)}{A(+,t)} = \frac{\sum_{k=1}^{\infty} (\tilde{D}_k - A_k)\, \mathbb{1}_{A_k \le t, L(A_k)>0}}{\sum_{k=1}^{\infty} \mathbb{1}_{A_k \le t, L(A_k)>0}},$$

is the total amount of remaining service times up to time t divided by the number of arrivals up to time t that see at least one job in the system. Thus, if the limit exists, we can *define*

$$\mathbb{E}[S_r \mid S_r > 0] = \lim_{t \to \infty} \frac{S_r(t)}{A(+,t)}.$$

Solution 2.14.2

1. 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., $n - 1$.

2.

$$\begin{aligned}
\mathbb{E}\left[L_Q\right] &= \sum_{n=1}^{\infty}(n-1)\pi(n) \\
&= (1-\rho)\sum_{n=1}^{\infty}(n-1)\rho^n \\
&= \rho(1-\rho)\sum_{n=1}^{\infty}(n-1)\rho^{n-1} \\
&= \rho\sum_{n=1}^{\infty}(n-1)\pi(n-1) \\
&= \rho\sum_{n=0}^{\infty}n\pi(n) \\
&= \rho\frac{\rho}{1-\rho}.
\end{aligned}$$

Another way to get the same result is by splitting: $\sum_{n=1}^{\infty} n\pi(n) - \sum_{n=1}^{\infty}\pi(n) = \mathbb{E}[L] - (1 - \pi(0)) = \mathbb{E}[L] - \rho$.

Solution 2.14.3 Because the remaining service time of the job in service, provided there is a job in the system upon arrival, is not exponentially distributed in general. Only for the $M/M/1$ queue the service times are exponentially distributed, hence memoryless. And only when the service time is memoryless, the service time after an interruption is still exponential.

Solution 2.14.4

1. The time in the system $\mathbb{E}[W]$ is the sum of the time in queue plus the service time of the job itself. Conditioning on the number of jobs N in the system at arrival moments, i.e., conditioning on $\pi(n)$, gives the result.

2. $\mathbb{E}\left[W_Q \,\middle|\, N=0\right] = 0$, of course. If $N = 1$, there must be a job in service. Because of the memoryless property of the service times, the remaining service time of the customer in service is still exponentially distributed with mean $\mathbb{E}[S]$. Thus, $\mathbb{E}\left[W_Q \,\middle|\, N=1\right] = \mathbb{E}[S]$.

3. The expected service time for a customer still in queue is $\mathbb{E}[S]$. Therefore, for each n we have that, when service times are exponential, $\mathbb{E}\left[W_Q \,\middle|\, N=n\right] = n\,\mathbb{E}[S]$. The probability to see n jobs in the system by an arrival is $\pi(n)$ which, by PASTA, is equal to $p(n)$. Since $\mathbb{E}[L] = \sum_n np(n)$, the result follows.

Solution 2.14.5 $\mathbb{E}\left[L_Q\right]$ is the time average of the queue length process (in steady state). Hence, $\mathbb{E}\left[L_q\right]$ contains also the time the queue is empty. $\mathbb{E}[L\,|\,B]$ the other hand, is the time average of the queue length process, provided the server is busy. Even though the queue may be empty while the server is busy, the fraction of time the queue length is zero given that the server is busy, is smaller than the fraction of time the queue length is zero. Hence, $\mathbb{E}[L\,|\,B] > \mathbb{E}\left[L_Q\right]$.

Solution 2.14.6 $\mathbb{V}[L] = \mathbb{E}\left[L^2\right] - (\mathbb{E}[L])^2$. We already know $\mathbb{E}[L]$ so it remains to compute $\mathbb{E}\left[L^2\right]$. With wolfram alpha we get

$$\begin{aligned}\mathbb{E}\left[L^2\right] &= \sum_{n=0}^{\infty} n^2 \pi(n) \\ &= \rho \frac{1+\rho}{(1-\rho)^2}.\end{aligned}$$

Thus,

$$\mathbb{V}[L] = \frac{\rho(1+\rho)}{(1-\rho)^2} - \frac{\rho^2}{(1-\rho)^2} = \frac{\rho}{(1-\rho)^2}.$$

To see how large this variance is, relative to the mean number of jobs in the system, we typically consider the square coefficient of variation (SCV). As $\mathbb{E}[L] = \rho/(1-\rho)$,

$$\frac{\mathbb{V}[L]}{(\mathbb{E}[L])^2} = \frac{1}{\rho}.$$

Thus, the SCV becomes smaller as ρ increases, but does not become lower than 1. So, realizing that the SCV of the exponential distribution is 1, the distribution of the number of jobs in the system has larger relative variability than the exponential distribution.

Lets see whether I can get the result for $\mathbb{E}\left[L^2\right]$ by myself. One way is to use the standard formula for a geometric series with $\rho < 1$:

$$\frac{1}{1-\rho} = \sum_{n=0}^{\infty} \rho^n.$$

If we differentiate the left and right hand side with respect to ρ and then multiply with ρ we obtain

$$\frac{\rho}{(1-\rho)^2} = \sum_{n=0}^{\infty} n\rho^n.$$

Again, differentiating and multiplying with ρ yields,

$$\begin{aligned}\rho \frac{(1-\rho)^2 + \rho 2(1-\rho)}{(1-\rho)^4} &= \rho \frac{1 - 2\rho + \rho^2 + 2\rho - 2\rho^2}{(1-\rho)^4} \\ &= \rho \frac{(1-\rho)^2}{(1-\rho)^4} \\ &= \rho \frac{1+\rho}{(1-\rho)^3} \\ &= \sum_{n=0}^{\infty} n^2 \rho^n\end{aligned}$$

and hence

$$(1-\rho) \sum_{n=0}^{\infty} n^2 \rho^n = \rho \frac{1+\rho}{(1-\rho)^2}$$

Observe that here we just compute the second moment of a geometric random variable.

Another way to derive the result is by noting that $\sum_{i=1}^{n} i = n(n+1)/2$ from which we get that $n^2 = -n + 2\sum_{i=1}^{n} i$. Substituting this in relation into $\sum_n n^2\rho^n$ leads to

$$\begin{aligned}
\sum_{n=0}^{\infty} n^2\rho^n &= \sum_{n=0}^{\infty}\left(\sum_{i=1}^{\infty} 2i\,\mathbb{1}_{i\le n} - n\right)\rho^n \\
&= \sum_{n=0}^{\infty}\sum_{i=0}^{\infty} 2i\,\mathbb{1}_{i\le n}\rho^n - \sum_{n=0}^{\infty} n\rho^n \\
&= \sum_{i=0}^{\infty} 2i\sum_{n=i}^{\infty}\rho^n - \frac{\mathbb{E}[L]}{1-\rho} \\
&= \sum_{i=0}^{\infty} 2i\rho^i\sum_{n=0}^{\infty}\rho^n - \frac{\mathbb{E}[L]}{1-\rho} \\
&= \frac{2}{1-\rho}\sum_{i=0}^{\infty} i\rho^i - \frac{\mathbb{E}[L]}{1-\rho} \\
&= \frac{2}{(1-\rho)^2}\mathbb{E}[L] - \frac{\mathbb{E}[L]}{1-\rho} \\
&= \frac{\mathbb{E}[L]}{1-\rho}\left(\frac{2}{1-\rho} - 1\right) \\
&= \frac{\mathbb{E}[L]}{1-\rho}\frac{1+\rho}{1-\rho} \\
&= \frac{\rho}{1-\rho}\frac{1+\rho}{(1-\rho)^2}.
\end{aligned}$$

A last method is based on z-transforms:

$$Z(z) = \mathbb{E}\left[z^L\right] = \sum_{n=0}^{\infty} z^n p(n) = (1-\rho)\sum_{n=0}^{\infty}(\rho z)^n = \frac{1-\rho}{1-\rho z}.$$

Then

$$\mathbb{E}[L] = \frac{d}{dz}Z(z)|_{z=1} = \frac{\rho}{1-\rho},$$

and

$$\mathbb{E}[L(L-1)] = \left(\frac{d}{dz}\right)^2 Z(z)\big|_{z=1} = \frac{2\rho^2}{(1-\rho)^2},$$

hence

$$\mathbb{E}[L]^2 = \mathbb{E}[L](L-1) + \mathbb{E}[L].$$

A bit of algebra gives the previous results.

Solution 2.14.7 Consider some state n and count all transitions that 'go in and out of' this state. Specifically, $A(n,t) + D(n-1,t)$ counts all transitions out of state n: $A(n,t)$ counts the number of arrivals that see n in the system upon arrival, hence immediately after such arrivals the system contains $n+1$ jobs; likewise, $D(n-1,t)$ counts all jobs that leave $n-1$ jobs behind, hence immediately before such jobs depart, the system contains n jobs. In a similar way, $A(n-1,t) + D(n+1,t)$ counts all transitions into state n. Therefore the diffence between the 'in'

transitions and the 'out' transitions is at most 1 for all t, so that we can write

$$\begin{aligned} A(n,t)+D(n-1,t) &\approx A(n-1,t)+D(n+1,t), \text{ hence} \\ \frac{A(n,t)+D(n-1,t)}{t} &\approx \frac{A(n-1,t)+D(n+1,t)}{t}, \text{ hence} \\ \frac{A(n,t)}{t}+\frac{D(n-1,t)}{t} &\approx \frac{A(n-1,t)}{t}+\frac{D(n+1,t)}{t}. \end{aligned}$$

Using the ideas of Section 2.9 this becomes for $t \to \infty$,

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

Since we are concerned here with the $M/M/1$ queue we have that $\lambda(n) = \lambda$ and $\mu(n) = \mu$, and using PASTA we have that $p(n) = \pi(n)$. We are done.

Solution 2.14.8 Since the service times are deterministic (and constant), I would guess that on average half of the service time remains at the moment a job arrives. If the service time is D always, then $\mathbb{E}[S_r \,|\, S_r > 0] = \lambda D/2$. We will prove this in Exercise 1.

Solution 2.14.9

1. When a problem is mainly of a computational type, I coded the solutions and show you all the numerical answers. Thus, please read the code too since this shows the precise steps by which I obtain the answer. (The code is typically nearly identical to the mathematical formulas; you should not have any difficulty understanding the code.) I also show many intermediate numerical results so that you can check each step in your computations. (In the computations below I typically use the simplest, but often not the most efficient, code.)

```
>>> P = [0.4, 0.3, 0.2, 0.05, 0.05]
>>> L = sum(n*P[n] for n in range(len(P)))
>>> L
1.05
```

   There can only be a queue when a job is in service. Since there is $m = 1$ server, we subtract m from the amount of jobs in the system. Before we do this, we need to ensure that $n-m$ does not become negative. Thus, $\mathbb{E}[L_Q] = \sum_n \max\{n-m,0\}p(n)$.

```
>>> m = 1
>>> Lq = sum(max(n-m,0)*P[n] for n in range(len(P)))
>>> Lq
0.45000000000000007
```

2.
```
>>> labda = 10./60
>>> Wq = Lq/labda # in minutes
>>> Wq
2.7000000000000006
>>> Wq/60 # in hours
0.04500000000000001
>>>
>>> W = L/labda # in minutes
```

```
>>> W
6.300000000000001
>>> W/60 # in hours
0.10500000000000001
```

3.

```
>>> from math import sqrt
>>> var_L = sum((n-L)**2*P[n] for n in range(len(P)))
>>> var_L
1.2475
>>> sqrt(var_L)
1.116915395184434
```

```
>>> var_Lq = sum((max(n-m,0)-Lq)**2*P[n] for n in range(len(P)))
>>> var_Lq
0.6475
>>> sqrt(var_Lq)
0.8046738469715541
```

4.

```
>>> mu = 1./(W-Wq)
>>> 1./mu # in minutes
3.5999999999999996
>>>
>>> rho = labda/mu
>>> rho
0.6
```

```
>>> rho = L-Lq
>>> rho
0.6
```

This checks the previous line.

The utilization must also by equal to the fraction of time the server is busy.

```
>>> u = 1 - P[0]
>>> u
0.6
```

Yet another way: Suppose we have m servers. If the system is empty, all m servers are idle. If the system contains one customer, $m-1$ servers are idle. Therefore, in general, the average fraction of time the server is idle is

$$1-u=\sum_{n=0}^{\infty}\max\{n-m,0\}p_n,$$

as in case there are more than m customers in the system, the number of idle servers is 0.

```
>>> idle = sum( max(m-n,0)*P[n] for n in range(len(P)))
>>> idle
0.4
```

Solution 2.14.10

1. $p_n = (1-\rho)\rho^n$
2.
```
>>> labda = 100. # per hour
>>> mu = 140. # per hour
>>> ES = 1./mu
>>> rho = labda/mu
>>> rho
0.7142857142857143
>>> 1-rho
0.2857142857142857
>>>
>>> L = rho/(1.-rho)
>>> L
2.5
>>> Lq = rho**2/(1.-rho)
>>> Lq
1.7857142857142858
>>>
>>> W = 1./(1.-rho) * ES
>>> W
0.024999999999999998
>>> Wq = rho/(1.-rho) * ES
>>> Wq
0.017857142857142856
```

Solution 2.14.11

1. $\mathbb{E}[W_Q] = \frac{1}{\lambda}\frac{\rho^2}{1-\rho} = \frac{\rho}{1-\rho}\mathbb{E}[S]$. Utilization is ρ. Hence, $\rho^2 = \lambda(1-\rho)$. Given $\mathbb{E}[W_Q]$ we can use this formula to solve for ρ with the abc-formula (and using that $\rho > 0$):

```
>>> labda = 5. # per minute
>>> Wq = 1.
>>> rho = -labda/2. + labda/2.*sqrt(1+4./labda)
>>> rho
0.8541019662496847
>>>
>>> ES = rho/labda
>>> ES
0.17082039324993695
```

2.
```
>>> Lq = labda*Wq
>>> Lq
5.0
>>>
>>> W = Wq + ES
>>> W
1.170820393249937
>>>
>>> L = labda*W
>>> L
5.854101966249685
```

3. The problem is to find n such that $\sum_{j=0}^{n} p_j > 0.9$.

```
P0 = 1.-rho
Pj = lambda j: rho**j*P0

tot = 0.
j = 0
while tot <= 0.9:
    tot += Pj(j)
    j += 1
```

Observe that j is one too high once the condition is satisfied.

```
>>> tot
0.9061042738330157
>>> j   # the number of chairs
15
```

As a check, I use that $p(0)\sum_{j=0}^{n} \rho^j = 1-\rho^{n+1}$.

```
>>> 1-rho**(j-1)
0.890064968964683
>>> 1-rho**j
0.9061042738330156
```

Solution 2.14.12 Follows right away from the hint.

Solution 2.14.13

1.
```
>>> labda = 6.
>>> mu = 5.
>>> r = labda/mu
>>> m = 1
>>> b = 2
```

Set $P = 1$ initially, and normalize later

```
from math import factorial

def computeP(n):
    if n <= m:
        return r**n/factorial(n)
    else:
        return r**n/(factorial(m)*m**(n-m))


>>> P = [1]*(m+b+1) # temporary value
>>> P
[1, 1, 1, 1]
>>>
>>> for n in range(1,m+b+1):
...     P[n] = computeP(n)
```

```
... tot = sum(P)
  File "< chunk 18 named None >", line 3
    tot = sum(P)
      ^
SyntaxError: invalid syntax
>>> tot
0.9061042738330157
>>>
>>> P = [p/tot for p in P] # normalize
>>> P
[1.103625740302256, 1.103625740302256, 1.103625740302256,
1.103625740302256]


>>> L = sum(n*P[n] for n in range(len(P)))
>>> L
6.621754441813536
>>>
>>> Lq = sum((n-m)*P[n] for n in range(m,len(P)))
>>> Lq
3.310877220906768
```

The number of jobs served per hour must be equal to the number of jobs accepted, i.e., not lost. The fraction of customers lost is equal to the fraction of customers that sees a full system.

```
>>> lost = labda*P[-1] # the last element of P
>>> lost
6.621754441813536
>>>
>>> accepted = labda*(1.-P[-1])
>>> accepted
-0.6217544418135361
```

2. Increase b to 5

```
>>> b = 5
>>> P = [1]*(m+b+1)
>>> P
[1, 1, 1, 1, 1, 1, 1]
>>>
>>> for n in range(1,m+b+1):
...     P[n] = computeP(n)
... tot = sum(P)
  File "< chunk 21 named None >", line 3
    tot = sum(P)
      ^
SyntaxError: invalid syntax
>>> tot
0.9061042738330157
>>>
>>> P = [p/tot for p in P] # normalize
>>> P
[1.103625740302256, 1.103625740302256, 1.103625740302256,
1.103625740302256, 1.103625740302256, 1.103625740302256,
```

```
1.1036257403022561]
>>>
>>> L = sum(n*P[n] for n in range(len(P)))
>>> L
23.176140546347376
>>>
>>> accepted = labda*(1.-P[-1])
>>> accepted
-0.6217544418135361
```

Solution 2.14.14 Lets just plot the waiting time for various values of λ.

```
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
sns.set_context("paper")

mu = 1.2 # per minute
t  = np.linspace(0, 10)

plt.figure(figsize=(6, 2))
for labda in np.linspace(0.5, 1., 10.):
    rho = labda/mu
    Ws = rho/(1.-rho)/labda
    res = 1 - rho + rho*(1.-np.exp(-1./Ws*t))
    plt.plot(t, res, label="%3.2f"%labda)
plt.legend()
filename = "figures/waiting_time.pdf"
plt.savefig(filename)
plt.close()
```

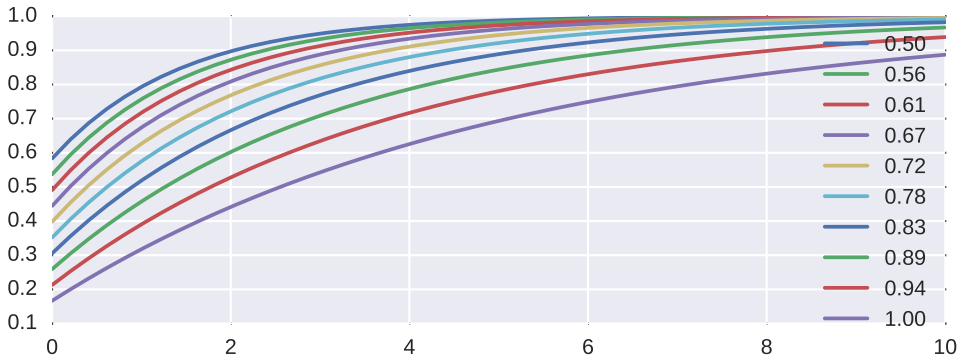


Figure 6.1: The waiting time distribution for various values of the arrival rate.

Solution 2.14.15

1.
```
>>> P = [0.4, 0.3, 0.2, 0.05, 0.05]
>>>
>>> m = 2
```

```
>>> Lq = sum((n-m)*P[n] for n in range(m,len(P)))
>>> Lq
0.15000000000000002
>>>
>>> L= sum(n*P[n] for n in range(len(P)))
>>> L
1.05
```

2.
```
>>> labda = 10./60
>>> Wq = Lq/labda # in minutes
>>> Wq
0.9000000000000001
>>> Wq/60 # in hours
0.015000000000000003
>>>
>>> W = L/labda
>>> W
6.300000000000001
```

3.
```
>>> from math import sqrt
>>> var_L = sum((n-L)**2*P[n] for n in range(len(P)))
>>> var_L
1.2475
>>> sqrt(var_L)
1.116915395184434
```

4.
```
>>> var_q = sum((max(n-m,0)-Lq)**2*P[n] for n in range(len(P)))
>>> var_q
0.22750000000000004
>>> sqrt(var_q)
0.4769696007084729
```

5.
```
>>> idle = sum( (m-n)*P[n] for n in range(m))
>>> idle
1.1
```

Solution 2.14.16

1.
```
>>> import numpy as np
>>> labda = [10., 15., 15., 10., 5.]
>>> m = 2
>>> mn = np.array([0,1,2,2,2])
>>> mu = 5*mn
>>> mu
array([ 0,  5, 10, 10, 10])


>>> P = [1]*5
>>> for i in range(1,5):
```

```
...      P[i] = labda[i-1]/mu[i]*P[i-1]
...
>>> P
[1, 2.0, 3.0, 4.5, 4.5]
>>> tot = sum(P)
>>> tot
15.0
>>> P = [p/tot for p in P]
>>> P # normed
[0.066666666666666666, 0.13333333333333333, 0.20000000000000001,
0.29999999999999999, 0.29999999999999999]
```

2.

```
>>> labdaBar = sum(labda[n]*P[n] for n in range(len(P)))
>>> labdaBar
10.166666666666666
>>>
>>>
>>> muBar = sum(mu[n]/mn[n]*P[n] for n in range(1,len(P)))/(1-P[0])
>>> muBar
4.9999999999999991
```

3.

```
>>> Ls = sum(n*P[n] for n in range(len(P)))
>>> Ls
2.6333333333333329
>>>
>>> m = 2
>>> Lq = sum((n-m)*P[n] for n in range(m,len(P)))
>>> Lq
0.89999999999999991
```

And now the waiting times:

```
>>> Ws = Ls/labdaBar
>>> Ws
0.25901639344262289
>>>
>>> Wq = Lq/labdaBar
>>> Wq
0.088524590163934422
```

To check my implementation of the code, I compute the results of the example on Page 144 of Hall's book.

```
>>> from math import factorial
>>>
>>> labda = 105./60
>>> mu = 120./60
>>> rho = labda/mu
>>> rho
0.875
>>> tot = sum(rho**n/factorial(n) for n in range(m)) +
rho**m/factorial(m)/(1.-rho/m)
>>> P0 = 1./tot
```

```
>>> P0
0.391304347826087
>>> Lq = rho**(m+1)/m/(factorial(m)*(1.-rho/m)**2)*P0
>>> Lq
0.2071256038647343
>>> Ls = Lq + rho
>>> Ls
1.0821256038647342
>>> Wq = Lq/labda
>>> Wq
0.11835748792270531
>>> Ws = Ls/labda
>>> Ws
0.6183574879227053
```

This checks.

Solution 2.14.17

1. Would one server/person do?

```
>>> labda = 28./60 # arrivals per minute
>>> ES = 4.
>>> rho = labda*ES
>>> rho
1.8666666666666667
```

Clearly not: The load is between 1 and 2 so we need at least two servers.

The remark that maintenance workers are compensated at the same rate as the tool crib workers confused me a bit at first. Some thought revealed that the consequence of this remark is that it just as expensive to let the tool crib workers wait (to help maintenance workers) as to let the maintenance workers wait for tools. (Recall, in queueing systems always somebody has to wait, either the customer or the server. If it is very expensive to let customers wait, the number of servers must be high, whereas if servers are relatively very expensive, customers have to do the waiting.)

```
def WQ(m, labda, ES):
    rho = labda*ES
    tot = sum(rho**n/factorial(n) for n in range(m)) +
rho**m/factorial(m)/(1.-rho/m)
    P0 = 1./tot
    Lq = rho**(m+1)/m/(factorial(m)*(1.-rho/m)**2)*P0
    return Lq/labda # Wq
```

Considering the scenario with one server is superfluous as $\rho > 1$ in that case.

```
>>> WQ(2, 28./60, 4)
27.034482758620694
>>> WQ(2, 28./60, 4)/60. # in hours
0.4505747126436782
>>> WQ(3, 28./60, 4)
1.3542675591474136
>>> WQ(3, 28./60, 4)/60. # in hours
```

```
0.022571125985790228
>>> WQ(4, 28./60, 4)
0.26778942672317646
>>> WQ(4, 28./60, 4)/60. # in hours
0.004463157112052941
```

2. It is not relevant to focus on the time in the system , as time in service needs to spent anyway. Hence, we focus on the waiting time in queue.

   Since both types of workers cost the same amount of money per unit time, it is best to divide the amount of waiting/idleness equally over both types of workers. I am inclined to reason as follows. The average amount of waiting time done by the maintenance workers per hour is $\lambda * Wq$. To see this, note that λ customers arrive, on average, each hour, and each customer waits on average W_q minutes. On the other hand, the total amount of minutes wasted by the tool crib employees is $m - \rho$, as $\rho = \lambda \mathbb{E}[S]$ is the average number of servers busy, while m crib servers are available.

   Now, while this makes perfect sense, to me at least, the formulas tell their own story. $\lambda W_q = L_q$. In other words, L_q is the average number of maintenance employees wasting their time in queue, while $m - \rho$ is the average number of crib workers wasting their time being idle. (To paraphrase Prince, doing something close to nothing.) Therefore, as both types of employees are equally expensive, we need to choose m such that $L_q \approx m - \rho$, where, of course, Lq depends on m.

```
>>> labda = 28./60
>>> ES = 4.
>>> rho = labda*ES
>>> m = 2
>>> labda*WQ(m, labda, ES)
12.616091954022991
>>> m-rho
0.1333333333333333
```

   Now the maintenance employees wait more than the tool crib employees.

```
>>> m = 3
>>> labda*WQ(m, labda, ES)
0.6319915276021264
>>> m-rho
1.1333333333333333
```

```
>>> m = 4
>>> labda*WQ(m, labda, ES)
0.12496839913748235
>>> m-rho
2.1333333333333333
```

   Clearly, $m = 3$ should do.

Solution 2.14.18

1.
```
>>> labda = 11.
>>> mu = 6
>>> m = 2
>>>
>>> rho = labda/mu
>>> rho < m  # is the system stable?
True
```

Compute Ls and so on

```
>>> tot = sum(rho**n/factorial(n) for n in range(m)) +
rho**m/factorial(m)/(1.-rho/m)
>>> P0 = 1./tot
>>> P0
0.043478260869565244
>>>
>>> Lq = rho**(m+1)/m/(factorial(m)*(1.-rho/m)**2)*P0
>>> Lq
9.64492753623188
>>> Lq/labda # Wq
0.8768115942028981
>>>
>>> Ls = Lq + rho
>>> Ls
11.478260869565213
>>> Ls/labda # Ws
1.0434782608695647
>>>
>>> ES = 1./mu # expected service time
>>> ES
0.16666666666666666
```

2.
```
>>> mu = m*6
>>>
>>> rho = labda/mu
>>>
>>> P0 = 1-rho
>>> P0
0.08333333333333337
>>>
>>> Lq = rho**2/(1.-rho)
>>> Lq
10.083333333333327
>>> Lq/labda # Wq
0.9166666666666661
>>>
>>> Ls = rho/(1.-rho)
>>> Ls
10.999999999999995
>>> Ls/labda # Ws
0.9999999999999996
>>>
>>> ES = 1./mu # expected service time
```

```
>>> ES
0.08333333333333333
```

It is of interest to compare the M/M/2 system to the M/M/1 system with twice the capacity.

Solution 2.14.19

Recall that for the $M/M/1$ queue, $\mathbb{E}[S_r \,|\, S_r > 0] = 1/\mu$. Applying Little's law to the queue, we have that $\lambda \mathbb{E}\left[W_Q\right] = EL_Q$, so that it follows that

$$\mathbb{E}\left[W_Q\right] = \rho \mathbb{E}[S_r \,|\, S_r > 0] + \mathbb{E}\left[L_Q\right] \mathbb{E}[S] = \frac{\rho}{\mu} + \rho \mathbb{E}\left[W_Q\right],$$

where we use that $\rho = \lambda \mathbb{E}[S]$. In words, the first term (at the right-hand side) is the probability to find a job in service, which is ρ, times the expected remaining service time if the server is busy, which is $1/\mu$ for exponentially distributed processing times. The second term is the probability to find the server busy, so that the arriving job has to wait, times the expected waiting time.

For the $M/M/c$ queue we now derive the corresponding expressions for this first and second term.

For the first, note that the probability that the job has to wait is the same as the probability that all servers are occupied at arrival. This is, clearly, $\sum_{k=c}^{\infty} p(k)$. If all servers are busy, the expected time until a service completion is $1/c\mu$, since there are c servers and the expected time to completion is $1/\mu$ for each individual server. Thus, the time until the first departure is the minimum of the remaining services at each of the servers. From Exercise 11 this has expection $1/c\mu$. Thus, for the $M/M/c$ queue

$$\rho \mathbb{E}[S_r \,|\, S_r > 0] = \sum_{k=c}^{\infty} p(k) \frac{1}{c\mu}.$$

For the second term, since there are c servers, jobs depart from the queue (hence enter service) at rate $c\mu$. The expected time in queue when there are n jobs in front, is therefore $n/c\mu$. Thus, on average, since the expected queue length is $\sum_{k=c}^{\infty}(k-c)p(k)$, we get

$$\mathbb{E}\left[L_Q\right] \frac{\mathbb{E}[S]}{c} = \sum_{k=c}^{\infty} p(k) \frac{k-c}{c\mu}.$$

All in all,

$$\begin{aligned} \mathbb{E}\left[W_Q\right] &= \rho \mathbb{E}[S_r \,|\, S_r > 0] + \mathbb{E}\left[L_Q\right] \frac{\mathbb{E}[S]}{c} \\ &= \sum_{k=c}^{\infty} p(k) \frac{1+k-c}{c\mu}. \end{aligned}$$

We can also give this a natural interpretation. How many jobs have to leave before the service of the arriving job can start? Well, the entire queue must be cleared plus one job in service. Thus, if there are $k-c$ job in queue, we have to wait for these jobs to finish, plus one of the jobs in service.

Observe also that, by PASTA, we have $\pi(k) = p(k)$, so that the arriving job also sees $p(k)$.

Solution 2.15.1 Jobs arrive at rate λ. The average number of items per job is $\mathbb{E}[B]$. Hence, work arrives as rate $\lambda\mathbb{E}[B]\mathbb{E}[S]$.This rate must be less than 1 for stability.

Solution 2.15.2

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

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

Solution 2.15.3 Start with the simple case, $B \equiv 2$. Then $\mathbb{V}[B] = 0$ and $\mathbb{E}[B] = 2$. The load is $\rho = \lambda\mathbb{E}[B]\mathbb{E}[S] = 1 \cdot 2 \cdot 25/60 = 5/6$. hence,

$$\mathbb{E}[L] = \frac{1}{2}\frac{5/6}{1/6}2 + \frac{1}{2}\frac{5/6}{1/6} = 5 + \frac{5}{2}.$$

Now the other case. $\mathbb{E}\left[B^2\right] = (1+4+9)/3 = 14/3$. Hence, $\mathbb{V}[B] = 14/3 - 4 = 2/3$. Hence,

$$C_s^2 = \frac{\mathbb{V}[B]}{(\mathbb{E}[B])^2} = \frac{2/3}{4} = \frac{1}{6}.$$

And thus,

$$\mathbb{E}[L] = \frac{1+1/6}{2}\frac{5/6}{1/6}2 + \frac{1}{2}\frac{5/6}{1/6} = \frac{7}{6}5 + \frac{5}{2}.$$

If we devide these two answers, we see that the ratio between $\mathbb{E}[L]$ for both answers is 10/9. In other words, we can reduce about 10% of the number of items in the system by working in fixed batch sizes.

Observe how easy it is with these models to get insight into the order of magnitude that can be achieved with changing work habits, such as working in contant batch sizes.

Observe also that its up to management to decide whether this reduction outweighs any efforts to reduce the variation in batch sizes. (In our model we assume that $B = 2$, i.e., we removed all variability. This is the best possible. Real life reductions will typically achieve less than the 10% reduction.)

Solution 2.15.4 Consider a queueing system with job arrival rate $\lambda = 6$ per hour and the jobs have batch sizes as indicated in the problem. The average number of items in the system follows like this

$$\mathbb{E}[B] = \frac{1+2+3+4}{4} = \frac{5}{2}$$
$$\mathbb{E}\left[B^2\right] = \frac{1+4+9+16}{4} = \frac{30}{4}$$
$$\mathbb{V}\left[B^2\right] = \frac{30}{4} - \frac{25}{4} = \frac{5}{4}$$
$$C_s^2 = \frac{5/4}{25/4} = \frac{1}{5}$$
$$\rho = \lambda\mathbb{E}[B]\mathbb{E}[S] = 6\frac{5}{2}\frac{1}{20} = \frac{3}{4}.$$

Hence,

$$\mathbb{E}[L] = \frac{1+1/5}{2}\frac{3/4}{1/4}\frac{5}{2} + \frac{1}{2}\frac{3/4}{1/4} = 6.$$

Thus, if the level is set to $r = 4$, then on average there will be two items short. Clearly, then, r should be at least 6. However, $\mathbb{E}[L]$ is just the average. Roughly speaking, in this case half of the demand will then be lost. So, if we take variability into account, r should be quite a bit bigger than 6. A more detailed analysis is necessary to determine the right value of r such that not more than a certain fraction of demand is lost. We will address this issue in Section 2.17.

Solution 2.15.5 Realize that this sort of problem is just a regular probability theory problem, nothing fancy. We use/adapt the tools you learned in calculus to carry out 2D integrals (or in this case 2D summations.)

$$\begin{aligned}\sum_{n=0}^{\infty} G(n) &= \sum_{n=0}^{\infty} \mathbb{P}\{B > n\} = \sum_{n=0}^{\infty}\sum_{i=n+1}^{\infty} \mathbb{P}\{B = i\} \\ &= \sum_{n=0}^{\infty}\sum_{i=0}^{\infty} 1\{n < i\}\mathbb{P}\{B = i\} = \sum_{i=0}^{\infty}\sum_{n=0}^{\infty} 1\{n < i\}\mathbb{P}\{B = i\} \\ &= \sum_{i=0}^{\infty} i\,\mathbb{P}\{B = i\} = \mathbb{E}[B].\end{aligned}$$

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

Solution 2.15.6

$$\begin{aligned}\sum_{i=0}^{\infty} iG(i) &= \sum_{i=0}^{\infty} i \sum_{n=i+1}^{\infty} \mathbb{P}\{B = n\} = \sum_{n=0}^{\infty} \mathbb{P}\{B = n\}\sum_{i=0}^{\infty} i1\{n \geq i+1\} \\ &= \sum_{n=0}^{\infty} \mathbb{P}\{B = n\}\sum_{i=0}^{n-1} i = \sum_{n=0}^{\infty} \mathbb{P}\{B = n\}\frac{(n-1)n}{2} \\ &= \sum_{n=0}^{\infty} \frac{n^2}{2}\mathbb{P}\{B = n\} - \frac{\mathbb{E}[B]}{2} = \frac{\mathbb{E}[B]^2}{2} - \frac{\mathbb{E}[B]}{2}.\end{aligned}$$

Solution 2.15.7

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

Solution 2.15.8

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

Solution 2.15.9 If $F(x) = 1 - e^{-\mu x}$, we obtain that

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

Solution 2.15.10 We have

$$\begin{aligned}\frac{\lambda}{\mu}\mathbb{E}\left[B^2\right] &= \frac{\lambda\,\mathbb{E}[B]}{\mu}\frac{\mathbb{E}\left[B^2\right]}{(\mathbb{E}[B])^2}\mathbb{E}[B] = \rho\frac{\mathbb{E}\left[B^2\right]}{(\mathbb{E}[B])^2}\mathbb{E}[B]\\ &= \rho\frac{(\mathbb{E}[B])^2 + \mathbb{V}[B]}{(\mathbb{E}[B])^2}\mathbb{E}[B] = \rho(1+C_s^2)\mathbb{E}[B].\end{aligned}$$

Solution 2.15.11 For the $M/M/1$ queue, each job contains just one item. Thus, $B \equiv 1$, hence $\mathbb{P}\{B=1\} = 1$, $\mathbb{E}\left[B^2\right] = \mathbb{E}[B] = 1$. Therefore, $\mathbb{E}[B_r(M/M/1)] = \rho$, and $\mathbb{E}[L(M/M/1)] = \rho/(1-\rho)$.

Such checks are always important to do. Does the new result reduce to the results of the models you know?

Solution 2.15.12

$$\frac{\mathbb{E}\left[L(M^X/M/1)\right]}{\mathbb{E}[L(M/M/1)]} = \frac{\mathbb{E}[B_r]}{\rho} = \frac{\mathbb{E}\left[B^2\right]}{2\mathbb{E}[B]} + \frac{1}{2}.$$

With this we can check whether this condition

$$1 \le \frac{\mathbb{E}\left[L(M^X/M/1)\right]}{\mathbb{E}[L(M/M/1)]} = \frac{\mathbb{E}\left[B^2\right]}{2\mathbb{E}[B]} + \frac{1}{2}$$

is always true. Clearly, it reduces to

$$\mathbb{E}[B] \le \mathbb{E}\left[B^2\right].$$

Next, we can use Jensen's inequality ($\phi(\mathbb{E}[X]) \le \mathbb{E}\left[\phi(X)\right]$ when ϕ is convex. In this case take $\phi(x) = x^2$.) to interpret this a bit further. First, by multiplying both sides with $\mathbb{E}[B]$ (to get $(\mathbb{E}[B])^2$ at the left hand side), we get

$$(\mathbb{E}[B])^2 \le \mathbb{E}\left[B^2\right]\mathbb{E}[B].$$

Since $B \ge 1$ (A job minimally contains one item) we have

$$(\mathbb{E}[B])^2 \le \mathbb{E}\left[B^2\right], \quad \text{Jensen's inequality} \le \mathbb{E}\left[B^2\right]\mathbb{E}[B], \quad B \ge 1.$$

Clearly, these inequalities are always satisfied. Hence,

$$1 \leq \frac{\mathbb{E}\left[L(M^X/M/1)\right]}{\mathbb{E}[L(M/M/1)]}$$

for all B.

In conclusion, if work arrives in batches, the average number of jobs in the system, increases, hence the average waiting time increases.

Solution 2.16.1 From Eq. (2.51) and the sentences above this equation, we have that $\mathbb{E}[S_r] = \rho\mathbb{E}[S_r \mid S_r > 0] = \lambda\mathbb{E}[S]\,\mathbb{E}[S_r \mid S_r > 0]$. Hence

$$\lambda\mathbb{E}[S_r \mid S_r > 0] = \frac{\mathbb{E}[S_r]}{\mathbb{E}[S]} = \lambda\frac{\mathbb{E}\left[S^2\right]}{2\mathbb{E}[S]}.$$

Solution 2.16.2 Use that

$$\frac{\mathbb{E}\left[S^2\right]}{(\mathbb{E}[S])^2} = \frac{(\mathbb{E}\left[S^2\right] - (\mathbb{E}[S])^2) + (\mathbb{E}[S])^2}{(\mathbb{E}[S])^2} = \frac{\mathbb{V}[S] + (\mathbb{E}[S])^2}{(\mathbb{E}[S])^2} = C_s^2 + 1.$$

Then

$$\mathbb{E}[S_r] = \frac{\lambda\mathbb{E}\left[S^2\right]}{2} = \frac{\mathbb{E}\left[S^2\right]\lambda(\mathbb{E}[S])^2}{2(\mathbb{E}[S])^2} = \frac{\mathbb{E}\left[S^2\right]}{2(\mathbb{E}[S])^2}\rho\mathbb{E}[S].$$

Solution 2.16.3 When S is deterministic, $\mathbb{P}\{S = 1/\mu\} = 1$. Therefore, $\mathbb{E}[S] = 1/\mu$ and $\mathbb{E}\left[S^2\right] = 1/\mu^2$.

Solution 2.16.4 When S is uniform, $\mathbb{E}[S] = \alpha^{-1}\int_0^\alpha x\,\mathrm{d}x = \alpha^2/2\alpha = \alpha/2$, where we take $\alpha = 2/\mu$ for notational convenience. Likewise $\mathbb{E}\left[S^2\right] = \alpha^{-1}\int_0^\alpha x^2\,\mathrm{d}x = \alpha^2/3$. Thus, $\mathbb{E}[S_r \mid S_r > 0] = \alpha/3 = 2/3\mu$.

Solution 2.16.5

$$\begin{aligned}
\mathbb{E}[S] &= \mu\int_0^\infty xe^{-\mu x}\,\mathrm{d}x = 1/\mu,\\
\mathbb{E}\left[S^2\right] &= \mu\int_0^\infty x^2e^{-\mu x}\,\mathrm{d}x = x^2e^{-\mu x}\big|_0^\infty + 2\int_0^\infty xe^{-\mu x}\,\mathrm{d}x\\
&= 2x/\mu e^{-\mu x}\big|_0^\infty + 2/\mu\int_0^\infty e^{-\mu x}\,\mathrm{d}x\\
&= \frac{2}{\mu^2}.
\end{aligned}$$

Solution 2.16.6

```
>>> labda = 5./60 # arrivals per minute
>>> a = 4.
>>> b = 6.
>>> ES = (a+b)/2.  # service time in minutes
>>> Var = (b-a)*(b-a)/12.
>>> SCV = Var/(ES**2)
```

```
>>>
>>> rho = labda*ES
>>>
>>> Wq = (1+SCV)/2.*rho/(1.-rho)*ES
>>> Wq # in minutes
1.8095238095238095
>>> Wq/60. # in hour
0.03015873015873016
>>>
>>>
>>> W = Wq + ES
>>> W
6.809523809523809
>>> Lq = labda*Wq
>>> Lq
0.15079365079365079
>>> L = labda*W
>>> L
0.5674603174603174
```

Solution 2.16.7 $\mathbb{E}\left[W_Q\right] = 4.5$ h. $\rho = \lambda\mathbb{E}[S] = (3/8)\cdot 2 = 3/4$.

One way to increase the capacity/reduce the average service time is to choose $\mathbb{E}[S] = 1$ hour and reduce C_s^2 to 1/4. Of course, there are many more ways. Reducing C_S^2 to zero is (nearly) impossible or very costly. Hence, ρ must go down, i.e., capacity up or service time down. Another possibility is to plan the arrival of jobs, i.e., reduce the variability in the arrival process. However, typically this is not possible. For instance, would you accept this as a customer?

Solution 2.16.8

The arrival process is assumed to be Poisson. There is also just one server. Hence, we can use the PK formula to compute the average queue length.

```
>>> labda = 20./60 # per hour
>>> ES = 2. # hour
>>> sigma = 1.
>>> SCV = sigma*sigma/(ES*ES)
>>> rho = labda*ES
>>> rho
0.6666666666666666
>>>
>>> Wq = (1+SCV)/2 * rho/(1-rho) * ES
>>> Wq
2.4999999999999996
>>> Lq = labda * Wq # Little's law
>>> Lq
0.8333333333333331
>>> W = Ws + ES
>>> W
2.6183574879227054
>>> L = labda * W
>>> L
0.8727858293075684


>>> ES = 2.1
>>> SD = 0.1
```

```
>>> SCV = SD**2/ES**2
>>> rho = labda*ES
>>> rho
0.7
>>>
>>> Lq = rho**2/(1.-rho)*(1.+SCV)/2.
>>> Lq
0.8185185185185182
>>> L = rho + Lq
>>> L
1.5185185185185182
```

Solution 2.16.9 Since the SCV for the exponential distribution is 1, hence $C_s^2 = 1$, we get

$$\mathbb{E}\left[W_Q\right] = \frac{1+1}{2}\frac{\rho}{1-\rho}\mathbb{E}[S] = \frac{\rho}{1-\rho}\mathbb{E}[S],$$

which we also found in the previous section.

Solution 2.16.10 For the $M/D/1$ the service time is deterministic.

Thus, the service time $S = T$ always; recall that S is deterministic under the assumptions in the exercise. Therefore, we must have that $\mathbb{P}\{S \le T\} = 1$. As an immediate consequence, $\mathbb{P}\{S > T\} = 1 - \mathbb{P}\{S \le T\} = 0$. Also, $\mathbb{P}\{S \le x\} = 0$ for all $x < T$. Thus, all probability mass is concentrated on the point T, thus, it is also impossible that the service time is smaller than T. (If you are into maths, then you should be aware of the fact that 'impossible' and 'almost surely' are not the same. Thus, my using of the word 'impossible' is not completely correct.)

To compute the SCV, I first compute $\mathbb{E}[S]$, then $\mathbb{E}\left[S^2\right]$ and then $\mathbb{V}[S] = \mathbb{E}\left[S^2\right] - (\mathbb{E}[S])^2$, and use the definition of coefficient of variation. I use these steps time and again.

First, $\mathbb{E}[S] = T$ since $\mathbb{P}\{S = T\} = 1$. We can also obtain this in a more formal way. The density $F(x)$ of S has a, so-called, atom at $x = T$, such $F(T) - F(T-) = 1$. Using this,

$$\mathbb{E}[S] = \int_0^\infty x \,\mathrm{d}F(x) = T \cdot (F(T) - F(T-)) = T.$$

Similarly

$$\mathbb{E}\left[S^2\right] = \int_0^\infty x^2 \,\mathrm{d}F(x) = T^2 \cdot (F(T) - F(T-)) = T^2.$$

Hence $\mathbb{V}[S] = \mathbb{E}\left[S^2\right] - (\mathbb{E}[S])^2 = 0$, hence $C_s^2 = 0$.

From the Pollackzek-Khintchine formula, the first term is $1 + C_s^2 = 1$ for the $M/D/1$ queue, and $1 + C_s^2 = 2$ for the $M/M/1$ queue. Hence,

$$\mathbb{E}\left[W_Q(M/D/1)\right] = \frac{\mathbb{E}\left[W_Q(M/M/1)\right]}{2}$$

thus,

$$\mathbb{E}\left[L_Q(M/D/1)\right] = \frac{\mathbb{E}\left[L_Q(M/M/1)\right]}{2}.$$

In both cases the expected service times are equal to T so that the loads are the same. Thus,

$$\begin{aligned}\mathbb{E}[L(M/D/1] &= \mathbb{E}\left[L_Q(M/D/1)\right] + \rho = \frac{\mathbb{E}\left[L_Q(M/M/1)\right]}{2} + \rho \\ &= \frac{\rho^2}{2(1-\rho)} + \rho = \frac{\rho(2-\rho)}{2(1-\rho)}.\end{aligned}$$

Solution 2.16.11

$$\begin{aligned}
\mathbb{E}[S] &= A/2,\\
\mathbb{E}\left[S^2\right] &= \int_0^A x^2\,\mathrm{d}x/A = A^2/3\\
\mathbb{V}[S] &= A^2/3 - A^2/4 = A^2/12\\
C_s^2 &= (A^2/12)/(A^2/4) = 1/3,\\
\rho &= \lambda A/2,\\
\mathbb{E}\left[W_Q\right] &= \frac{1+C_s^2}{2}\frac{\lambda A/2}{1-\lambda A/2}\frac{A}{2},\\
\mathbb{E}[W] &= \mathbb{E}\left[W_Q\right] + \frac{A}{2}\\
\mathbb{E}[L] &= \lambda\mathbb{E}[W].
\end{aligned}$$

Solution 2.16.12 Since the interarrival times are memoryless, the expected time to the first arrival after a busy time stops is also $\mathbb{E}[X] = 1/\lambda$.

Solution 2.16.13 Since $\rho = \lambda\mathbb{E}[S]$, $\mathbb{E}[I] = 1/\lambda$ and from Exercise 6

$$\mathbb{E}[B] = \frac{\rho}{1-\rho}\mathbb{E}[I] = \frac{\lambda\mathbb{E}[S]}{1-\rho}\frac{1}{\lambda}.$$

The result follows.

Solution 2.16.14 The system can contain at most 1 job. Necessarily, if the system contains a job, this job must be in service. All jobs that arrive while the server is busy are blocked, in other words, rejected. Just after a departure, the average time until the next arrival is $1/\lambda$, then a service starts with an average duration of $\mathbb{E}[S]$. After this departure, a new cycle start. Thus, the utilization is $\mathbb{E}[S]/(1/\lambda + \mathbb{E}[S]) = \lambda\mathbb{E}[S]/(1+\lambda\mathbb{E}[S])$. Since not all jobs are accepted, the utilization ρ cannot be equal to $\lambda\mathbb{E}[S]$.

Solution 2.16.15 Let $p(0)$ be the fraction of time the server is idle. By PASTA, $\pi(0) = p(0)$. Thus, the fraction of accepted jobs is $\lambda\pi(0)$. Therefore, the departure rate $\gamma = \pi(0)\lambda$. The loss rate is $\lambda - \gamma = (1-\pi(0))\lambda$.

Since $\lambda\pi(0)$ is the rate at which jobs enter the system, the load must be $\lambda\pi(0)\mathbb{E}[S]$. Since the load is also $1-\pi(0)$, it follows from equating this, that $\lambda\mathbb{E}[S]\pi(0) = 1-\pi(0)$, from which $\pi(0) = 1/(1+\lambda\mathbb{E}[S])$, hence $1-\pi(0) = \lambda\mathbb{E}[S]/(1+\lambda\mathbb{E}[S])$, which is the same an in the previous problem.

Note that $\gamma < \lambda$, as it should be the case.

Solution 2.16.16 Typically, in the $G/G/1$ queue, the arrivals do not see time-averages. Consequently, the fraction of arrivals that are blocked is not necessarily equal to the utilization ρ.

Again, take jobs with duration 59 minutes and interarrival times of 1 hour. The load is 59/60, but no job is lost, also not in the $G/G/1/1$ case. Thus, $\gamma = \lambda$ in this case.

Solution 2.17.1 The left hand side of Eq. (2.63) is identical, so we only have to concentrate on the right hand side. In the $M/M/1$ queue, all batches have size 1. Thus, $\mathbb{P}\{B=1\} = f(1) = 1$ and $f(k) = 0$ for $k \neq 1$. Thus, $G(0) = 1$ and $G(1) = G(2) = \cdots = 0$. Thus, $\sum_{m=0}^{n} G(n-m)p(m) = G(0)p(n) = p(n)$.

Solution 2.17.2 $A(n,n,t)$ counts all jobs up to time t that see n items and bring at least $n-n+1=1$ unit of work. As each job brings at least 1 item, $A(n,n,t)$ counts all jobs that see n items at arrival.

Solution 2.17.3 Because arrivals do not occur in single units, but in batches.

Solution 2.17.4 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 2.15 to see that

$$
\begin{aligned}
\mu\mathbb{E}[L] &= \sum_{n=0}^{\infty} n\,\mu\pi(n), \text{now substitute for } \mu\pi(n) \text{ the above recursion,} \\
&= \lambda \sum_{n=0}^{\infty} n \sum_{i=0}^{n-1} \pi(i)G(n-1-i) = \lambda \sum_{n=0}^{\infty} n \sum_{i=0}^{\infty} 1\{i<n\}\pi(i)G(n-1-i) \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} 1\{i<n\} nG(n-1-i) = \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=i+1}^{\infty} nG(n-1-i) \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} (n+i+1)G(n) = \lambda \sum_{i=0}^{\infty} \pi(i) \left[\sum_{n=0}^{\infty} nG(n) + (i+1) \sum_{n=0}^{\infty} G(n) \right] \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \sum_{n=0}^{\infty} nG(n) + \lambda\mathbb{E}[B] \sum_{i=0}^{\infty} \pi(i)(i+1) \\
&= \lambda \sum_{i=0}^{\infty} \pi(i) \frac{\mathbb{E}[B]^2 - \mathbb{E}[B]}{2} + \lambda\mathbb{E}[B](\mathbb{E}[L]+1) \\
&= \lambda \frac{\mathbb{E}[B]^2 - \mathbb{E}[B]}{2} + \lambda\mathbb{E}[B]\,\mathbb{E}[L] + \lambda\mathbb{E}[B] \\
&= \lambda \frac{\mathbb{E}[B]^2}{2} + \lambda\mathbb{E}[B]\,\mathbb{E}[L] + \lambda \frac{\mathbb{E}[B]}{2},
\end{aligned}
$$

Solution 2.17.5

Let's first deal with complete rejection. Suppose a batch of size k arrives when the system contains n jobs. When $k+n \leq K$, the batch can be accepted since the entire batch will fit into the queue. When, however, $k+n > K$, the batch has to be rejected.

Now consider an imaginary line between states with n and $n+1$ jobs in the system. This imaginary line separates the state space into two disjoint part: one part with states $0,1,\ldots,n$ and the other part with states $n+1,n+2,\ldots,K$. We call this imaginary line 'level n'. We call an 'upcrossing' a transition from some state $m \leq n$ to a state $l \geq n+1$. We also say in that case that level n is 'crossed from below', i.e., from some state $m \leq n$ to a state $l \geq n+1$. Likewise, a 'downcrossing' of level n happens when this level is crossed from 'above', i.e. from some state $l \geq n+1$ to some state $m \leq n$.

If the system contains n jobs, level n is crossed from below with rate $\lambda\pi(n)\mathbb{P}\{B \leq K-n\}$. More generally, when the system is in state $m \leq n$, we need a batch of at least $n+1-m$ to cross the n. Moreover, any batch larger than $K+1-m$ gets rejected. Thus, when the system contains

$m \le n$ jobs, the rate at which level n is crossed from below is

$$\lambda\pi(m)\mathbb{P}\{n+1-m \le B \le K+1-m\} = \lambda\pi(m)[G(n-m)-G(K-m)].$$

Since the server serves only single items, level n can only be crossed from above from state $n+1$. This happens at rate $\mu p(n+1)$.

Finally, since on the long run the number of up- and crossings must be the same, the up- and downcrossing rates must match. This implies that the balance equations becomes

$$\mu\pi(n+1) = \lambda \sum_{m=0}^{n} \pi(m)[G(n-m)-G(K-m)],$$

for $n = 0,\ldots,K$.

Before we continue with the other acceptance rules, it is important to check this result. In general, off-by-one errors are easily, and commonly, made, so we need to test the above on simple cases.

- If $K \to \infty$, then $G(K-m) = \mathbb{P}\{B > K-m\} \to 0$, so we get our earlier result.
- Take $n = K$. Then $G(n-m)-G(K-m) = 0$ for all m. Then the right hand side is 0, as it should.
- Taken $n = 0$ and $K = 1$, then $\mu\pi(1) = \lambda\pi(0)$. This also makes sense.
- Take n much smaller than K. If the batch size is maximally 2, then for small n the entire batch must fit. Let's see if this holds in the above formula. If n much smaller than K, then also m is much smaller than K (since in the right hand size, $m \le n$. But then $G(K-m) \le G(2) = 0$, as it should. (Observe that G is a decreasing function of it argument; its a survival function.)

The complete-acceptance policy is actually quite simple. As any batch will be accepted when $n \le K$, the queue length is not bounded. Only when the number of jobs in the system is larger than K, we do not accept jobs.

$$\mu\pi(n+1) = \begin{cases} \lambda \sum_{m=0}^{n} \pi(m)G(n-m), & \text{for } n \le K, \\ \lambda \sum_{m=0}^{K} \pi(m)G(n-m), & \text{for } n > K. \end{cases}$$

For the partial acceptance case, any job is accepted, but the system only admits whatever fits. Thus, the rate at which level $n < K$ is crossed is not affected by this acceptance policy, and therefore,

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

for $n = 0,1,\ldots,K$.

Solution 2.17.6 The recursion for n becomes $\pi(n) = \lambda/\mu \sum_{i=0}^{n-1} \pi(n-1-i)G(i)$. Since $G(i) = 0$ for $i \ge 3$ we rewrite this to

$$\pi(n) = \lambda/\mu \sum_{i=0}^{\min(\{n-1,l\}} \pi(n-1-i)G(i),$$

where $l = 3$.

In python this reads:

```
p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, l)))
```

The following code carries out the recursion for the $M/M/1$ queue and compares the result to the (unnormalized) probabilities $(\lambda/\mu)^n$. Thus, we can test the code right away.

```
import numpy as np

l = 1 # M/M/1 has batch size 1
f = np.ones(l + 1) / l # f[0] =1, f[1] = 1
f[0] = 0 # set f[0] = 0
F = np.cumsum(f) # distribution
G = np.ones_like(F) - F # survivor function

labda = 1
mu = 3
num = 30
p = np.ones(num)

for n in range(1, num):
    p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, l)))
    print(n, p[n], (labda / mu)**n)
```

```
1 0.333333333333 0.3333333333333333
2 0.111111111111 0.1111111111111111
3 0.037037037037 0.03703703703703703
4 0.0123456790123 0.012345679012345677
5 0.00411522633745 0.004115226337448558
6 0.00137174211248 0.0013717421124828527
7 0.000457247370828 0.00045724737082761756
8 0.000152415790276 0.0001524157902758725
9 5.08052634253e-05 5.0805263425290837e-05
10 1.69350878084e-05 1.693508780843028e-05
11 5.64502926948e-06 5.645029269476759e-06
12 1.88167642316e-06 1.8816764231589195e-06
13 6.27225474386e-07 6.272254743863065e-07
14 2.09075158129e-07 2.090751581287688e-07
15 6.96917193763e-08 6.969171937625627e-08
16 2.32305731254e-08 2.3230573125418753e-08
17 7.74352437514e-09 7.743524375139585e-09
18 2.58117479171e-09 2.5811747917131946e-09
19 8.60391597238e-10 8.603915972377315e-10
20 2.86797199079e-10 2.867971990792438e-10
21 9.55990663597e-11 9.559906635974793e-11
22 3.18663554532e-11 3.186635545324931e-11
23 1.06221184844e-11 1.0622118484416435e-11
24 3.54070616147e-12 3.540706161472145e-12
25 1.18023538716e-12 1.1802353871573816e-12
26 3.93411795719e-13 3.934117957191272e-13
27 1.3113726524e-13 1.3113726523970905e-13
28 4.37124217466e-14 4.3712421746569684e-14
29 1.45708072489e-14 1.457080724885656e-14
```

The test is convincing.

```
l = 3
f = np.ones(l + 1) / l
```

```
f[0] = 0
F = np.cumsum(f)
G = np.ones_like(F) - F
num = 30
p = np.ones(num)
for n in range(1, num):
    p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, l)))
    print(n, p[n])
```

```
1 0.333333333333
2 0.333333333333
3 0.296296296296
4 0.20987654321
5 0.172839506173
6 0.137174211248
7 0.107453132144
8 0.0855052583448
9 0.0676218056191
10 0.053481007299
11 0.0423546546089
12 0.0335164204493
13 0.0265266197627
14 0.0209963727551
15 0.0166176420267
16 0.013152476817
17 0.0104098985844
18 0.00823914349042
19 0.006521078273
20 0.00516126893158
21 0.00408501187013
22 0.00323318352739
23 0.00255898258378
24 0.0020253696307
25 0.00160302862078
26 0.00126875641195
27 0.00100418845646
28 0.000794789646005
29 0.000629055806986
```

Stopping at 30 seems ok. The last step is normalize.

```
p /= p.sum()  # normalize
for n in range(1, num):  # print normalized results
    print(n, p[n])
```

```
1 0.111199612316
2 0.111199612316
3 0.098844099836
4 0.0700145707172
5 0.0576590582377
6 0.0457611573315
7 0.0358462399097
8 0.0285244547366
9 0.0225585557067
10 0.0178412018337
11 0.0141294635168
12 0.0111810388805
```

```
13 0.00884924950098
14 0.00700436553119
15 0.00554362605289
16 0.00438765096912
17 0.00347273006048
18 0.00274856868584
19 0.00217542412751
20 0.00172179331274
21 0.00136275520879
22 0.00107858626437
23 0.000853673613716
24 0.000675660953187
25 0.000534768483485
26 0.000423255663394
27 0.000334996101149
28 0.000265140901524
29 0.000209852285585
```

Now that we have the probabilities we can do all experiments we like.

```
>>> EL = sum(n*p[n] for n in range(len(p)))
>>> EL
3.309069824793065
```

It is of interest to compare this result to the equation above Eq. (2.56).

```
>>> EB = sum(k*fk for k,fk in enumerate(f))
>>> EB2 = sum(k*k*fk for k,fk in enumerate(f))
>>> EB2
4.6666666666666661
>>> rho=labda*EB/mu
>>> RHS = labda/mu*EB2/2 + rho/2
>>> EL=RHS/(1-rho)
>>> EL
3.3333333333333326
```

So, after all, stopping at 30 seems not quite ok: there is a slight difference between the two expectations. Lets run the recursion up to 100, and see what we get then.

```
num = 100
p = np.ones(num)
for n in range(1, num):
    p[n] = labda / mu * sum(p[n - 1 - i] * G[i] for i in range(min(n, l)))
p /= p.sum()  # normalize
EL = sum(n*p[n] for n in range(len(p)))
print(EL)

3.33333332712
```

This does the job.

As a last step, I want to check Eq. (2.56).

```
>>> VB = EB2 - EB*EB
>>> C2 = VB/EB/EB
>>> EL = (1+C2)/2*rho/(1-rho)*EB + rho/(1-rho)/2
>>> EL
3.3333333333333326
```

The algebra is, most probably, correct. I spent at least one hour to get Eq. (2.56) correct. I am now, also numerically, convinced that all results are ok.

Solution 2.18.1 Define, for the first term,

$$D(0,n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \le t}\, \mathbb{1}_{L(D_{k-1})=0}\, \mathbb{1}_{Y_k > n}$$

Then

$$\begin{aligned}
\lim_{t\to\infty} \frac{D(0,n,t)}{t} &= \lim_{t\to\infty} \frac{D(t)}{t} \frac{D(0,t)}{D(t)} \frac{D(0,n,t)}{D(0,t)} \\
&= \gamma\delta(0) \lim_{t\to\infty} \frac{D(0,n,t)}{D(0,t)} \\
&= \gamma\delta(0)\mathbb{P}\{Y > n\} \\
&= \gamma\delta(0)G(n).
\end{aligned}$$

For the second term, define

$$D(m,n,t) = \sum_{k=1}^{\infty} \mathbb{1}_{D_k \le t}\, \mathbb{1}_{L(D_{k-1})=m}\, \mathbb{1}_{Y_k > n-m+1}$$

then

$$\begin{aligned}
\lim_{t\to\infty} \frac{D(m,n,t)}{t} &= \lim_{t\to\infty} \frac{D(t)}{t} \frac{D(m,t)}{D(t)} \frac{D(m,n,t)}{D(m,t)} \\
&= \gamma\delta(m) \lim_{t\to\infty} \frac{D(m,n,t)}{D(m,t)} \\
&= \gamma\delta(m)\mathbb{P}\{Y > n-m+1\} \\
&= \gamma\delta(m)G(n-m+1).
\end{aligned}$$

Solution 2.18.2 When $L(D_{k-1}) > 0$, job k is already in the system when job $k-1$ finishes its service and leaves. Thus, at the departure time D_{k-1} of job $k-1$, the service of job k can start right away at D_{k-1}. Then, $D_k = D_{k-1} + S_k$.

When job $k-1$ leaves an empty system behind, $D_k = A_k + S_k$, since job k sees an empty system, hence its service can start right away after its arrival time at time A_k. Since the arrival process is Poisson by assumption, the time to the next arrival after D_{k-1} is exponentially distributed with rate λ. (Recall the memoryless property of the interarrival times.) Thus, $A_k - D_{k-1}$ has the same distribution as X_k, so that $\mathbb{P}\{D_k - D_{k-1} \le x\} = \mathbb{P}\{X_k + S_k \le x\}$. BTW: Can you simplify this expression by using conditioning on X_k?

Solution 2.18.3 If the service time is s units long, we need to know the number of arrivals during this interval. By assumption, the arrival process is Poisson.

Solution 2.18.4 We use a conditioning argument to arrive at this result. The probability that the service time is s units long is written like $\mathrm{d}F(s) = \mathbb{P}\{S \in \mathrm{d}s\}$. (To properly define this we need measure theory...). Thus $\mathbb{P}\{Y_k = j\} = \int_0^\infty \mathbb{P}\{Y_k = j \,|\, S = s\}\, \mathbb{P}\{S \in \mathrm{d}s\} = \int_0^\infty \mathbb{P}\{Y_k = j \,|\, S = s\}\, \mathrm{d}F(s)$. Using the answer of the previous problem we arrive at the result.

Solution 2.18.5 If $\mathbb{P}\{S = s\} = 1$, then $\int_0^\infty g(x)\,\mathrm{d}F(x) = g(s)$ for any sensible function g. More specifically, since $\mathbb{P}\{S = s\} = 1$, the distribution F has an atom at s. The size of the atom is $F(s) - F(s-) = 1$. Thus, for instance, $\int_0^\infty x\,\mathrm{d}F(x) = s(F(s) - F(s-)) = s$. The function x in this integral can be replaced by any function $g(x)$ (provided g satisfies some technical regularity conditions. If you are mathematically interested: it must be measurable.) Applying this result to the function $e^{-\lambda x}(\lambda x)^j/j!$ we obtain the answer.

Solution 2.18.6 Use conditional probability to see that

$$\begin{aligned}\mathbb{P}\{Y_n = j\} &= \int_0^\infty e^{-\lambda s}\frac{(\lambda s)^j}{j!}\,\mathrm{d}F(s) = \int_0^\infty e^{-\lambda s}\frac{(\lambda s)^j}{j!}\mu e^{-\mu s}\,\mathrm{d}s \\ &= \frac{\mu}{j!}\lambda^j\int_0^\infty e^{-(\lambda+\mu)s}s^j\,\mathrm{d}s = \frac{\mu}{j!}\left(\frac{\lambda}{\lambda+\mu}\right)^j\int_0^\infty e^{-(\lambda+\mu)s}((\lambda+\mu)s)^j\,\mathrm{d}s \\ &= \frac{\mu}{j!}\left(\frac{\lambda}{\lambda+\mu}\right)^j\frac{j!}{\lambda+\mu}.\end{aligned}$$

Here I also used the hints and results of Exercise 8.

In hindsight, this result could have been derived in another way, in fact by using the result of Exercise 12 in which we analyzed a merged Poisson process. Consider the Poisson process with rate $\lambda+\mu$ that arises when the arrival and service process are merged. The probability that an arrival corresponds to an epoch of the merged process is $\lambda/(\lambda+\mu$ and the probability that a departure corresponds to an epoch of the merged process is $\mu/(\lambda+\mu)$. The probability that j arrivals occur before a service occurs, is the same as the probability that a geometrically distributed random variable with success probability $\mu/(\lambda+\mu) = 1-p$ takes the value j.

Solution 2.18.7 Take $\alpha = \lambda/(\lambda+\mu)$ so that $f(j) = (1-\alpha)\alpha^j$.

$$\begin{aligned}G(j) &= \sum_{k=j+1}^\infty f(k) = (1-\alpha)\sum_{k=j+1}^\infty \alpha^k \\ &= (1-\alpha)\sum_{k=0}^\infty \alpha^{k+j+1}, \text{ by change of variable} \\ &= (1-\alpha)\sum_{k=0}^\infty \alpha^k\alpha^{j+1} = (1-\alpha)\alpha^{j+1}\sum_{k=0}^\infty \alpha^k \\ &= (1-\alpha)\alpha^{j+1}\frac{1}{1-\alpha} = \alpha^{j+1}.\end{aligned}$$

Solution 2.18.8 Take $n = 0$. Then $f(0)\pi(1) = (1-\alpha)\pi(1)$, and $\pi(0)G(0) = \pi(0)\alpha$. Thus, $\pi(1) = \pi(0)\alpha/(1-\alpha) = \rho\pi(0)$.

For $n = 1$,

$$\begin{aligned}(1-\alpha)\pi(2) &= \pi(0)G(1) + \pi(1)G(1) = \pi(0)G(1)(1+\rho) \\ &= \pi(0)\alpha^2(1+\rho) = \pi(0)\alpha\alpha(1+\rho) = \pi(0)\alpha\rho.\end{aligned}$$

Dividing by $1-\alpha$, we get

$$\pi(2) = \pi(0)\rho^2$$

Finally, now that we suspect that $\pi(n) = \rho^n \pi(0)$, let's fill it in, and see whether it checks. We divide both sides by $\pi(0)$ so that we are left with checking that

$$\begin{aligned}
(1-\alpha)\rho^{n+1} &= \alpha^{n+1} + \sum_{m=1}^{n} \rho^m \alpha^{n-m+2} \\
&= \alpha^{n+1} + \alpha^{n+2} \sum_{m=1}^{n} (\rho/\alpha)^m \\
&= \alpha^{n+1} + \alpha^{n+1}\rho \sum_{m=0}^{n-1} (\rho/\alpha)^m \\
&= \alpha^{n+1} + \alpha^{n+1}\rho \sum_{m=0}^{n-1} (\rho/\alpha)^m \\
&= \alpha^{n+1} + \alpha^{n+1}\rho \frac{1-(\rho/\alpha)^n}{1-\rho/\alpha} \\
&= \alpha^{n+1} - \alpha^{n+1}(1-(\rho/\alpha)^n), \quad \text{as } \rho/\alpha = 1+\rho, \\
&= \alpha^{n+1}(\rho/\alpha)^n = \alpha\rho^n.
\end{aligned}$$

Since $\rho = \alpha/(1-\alpha)$ we see that the left and right hand sides are the same.

Thus we get that $\pi(n) = \delta(n) = (1-\rho)\pi(0)$; a result we obtained earlier for the $M/M/1$ queue.

Solution 2.18.9 A simple numerical method is as follows. Make a grid of size $\mathrm{d}x$, for some small number $\mathrm{d}x$, e.g. $\mathrm{d}x = 1/100$, and write $f_i = \mathbb{P}\{S \in (i\,\mathrm{d}x, (i+1)\,\mathrm{d}x]\} = F((i+1)\,\mathrm{d}x) - F(i\,\mathrm{d}x)$. Then

$$\mathbb{P}\{Y_k = j\} = \int_0^\infty e^{-\lambda s} \frac{(\lambda s)^j}{j!}\,\mathrm{d}F(s) \approx \sum_{i=1}^{\infty} e^{-\lambda i\,\mathrm{d}x} \frac{(\lambda i\,\mathrm{d}x)^j}{j!} f_i\,\mathrm{d}x$$

Lets try a numerical experiment. Topics like these are, of course, not necessary for the course, but I hope that you study it nonetheless. You will have to use it for other courses later anyway.

```
import numpy as np

labda = 3
mu = 4
j = 5
dx = 1 / 100


def F(x):
    return 1 - np.exp(-mu * x)


def f(x):
    return F(x + dx) - F(x)


def term(i):
    res = np.exp(-labda * i * dx)
    res *= (labda * i * dx)**j / np.math.factorial(j)
    res *= f(i*dx) * dx
    return res
```

```
>>> print(sum(term(i) for i in range(50)))
1.11588350072e-05
>>> print(sum(term(i) for i in range(500)))
8.09880771006e-05
>>> print(sum(term(i) for i in range(5000)))
8.09880771273e-05
```

Since I don't know when to stop the integral I just try a few values; of course stopping the integration at $x = 50$ is too small, since $50\,\mathrm{d}x = 50/100 = 1/2$, but I include it for illustrative purposes. Stopping at 500 seems ok, since the results of the last two integrals are nearly the same. This also suggeststst that $\mathrm{d}x = 1/100$ is sufficiently small. In general, however, one must take care and try various values for $\mathrm{d}x$ and the integration limits.

In general, for more complicated situations is it best to use a numerical library to compute the above integral. The methods have been designed to produce good and reliable results, and, typically, it is very hard to improve these methods. Thus, let's try a real number cruncher.

```
>>> from scipy.integrate import quad
>>>
>>> def g(x):
...     return np.exp(-labda*x) * (labda*x)**j/np.math.factorial(j) * f(x)
...
>>> print(quad(g, 0, np.inf))
(8.098807712760667e-05, 3.4086429822163874e-11)
```

Comparing these results to our simple estimates we see that they are the same.

Solution 2.19.1 Starting from the distribution function $\mathbb{P}\{S \le x\}$, we define

$$\mathbb{P}\left\{X^{(n)} = i\right\} = \mathbb{P}\{S \le (i+1)/n\} - \mathbb{P}\{S \le i/n\} = \mathbb{P}\{S \in (i/n, (i+1)/n]\}\,.$$

Solution 2.19.2 The time it takes to serve one unit is $1/n$. Thus, the fraction of time in service in the batch queue becomes negligible compared to the time in queue. Another way to see this, is to use

$$\frac{\mathbb{E}\left[L^{(n)}\right]}{n} = \frac{\mathbb{E}\left[L_Q^{(n)}\right]}{n} + \frac{\mathbb{E}\left[L_s^{(n)}\right]}{n}.$$

Since $L_s^{(n)} \le 1$, $\mathbb{E}\left[L^{(n)}\right]/n \approx \mathbb{E}\left[L_Q^{(n)}\right]/n$ for n large.

Solution 2.20.1 If $c = 1$, as is the case for the $M/M/1$ queue, $\sqrt{2(c+1)} - 1 = 2 - 1 = 1$, so that (2.73) reduces to $\mathbb{E}\left[W_Q\right] = \rho/(1-\rho)\mathbb{E}[S]$. Recall that $C_a^2 = C_s^2 = 1$ for the $M/M/1$.

Solution 2.20.2 Not necessarily. Since jobs do not arrive in general as a Poisson process, we cannot use the PASTA property to conclude that the time average queue length $\mathbb{E}\left[L_Q\right]$ is the same as the average queue length as observed by customers.

(Can you provide an example that shows this difference? Hint, we did this earlier.)

Solution 2.20.3 This follows from the observation that $\lambda(1-\beta)$ is the net arrival rate, as jobs are lost at a rate $\lambda\beta$. Hence, the load must be $\lambda(1-\beta)/\mu$. As the load must be equal to ρ, it follows that $\rho = \lambda(1-\beta)/\mu$, from which the other result immediately follows.

Solution 2.20.4

1. $\rho = \lambda\mathbb{E}[S] = 1/60 \cdot 50 = 5/6$. 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 $\mathbb{E}[L] = 1 \cdot 5/6 = 5/6$. 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 $\mathbb{E}\left[W_Q\right] = 0$ since $\mathbb{E}\left[L_Q\right] = 0$. Interestingly, we get the same from Kingman's formula.

2. Again $\mathbb{E}[S]$ is 50 seconds, so that $\rho = 5/6$. Also $C_a^2 = 0$. For the C_S^2 we have to do some work.

$$\begin{aligned}
\mathbb{E}[S] &= \frac{20}{2} + \frac{80}{2} = 50 \\
\mathbb{E}\left[S^2\right] &= \frac{400}{2} + \frac{6400}{2} = 3400 \\
\mathbb{V}[S] &= \mathbb{E}[S]^2 - (\mathbb{E}[S])^2 = 3400 - 2500 = 900 \\
C_S^2 &= \frac{\mathbb{V}[S]}{(\mathbb{E}[S])^2} = \frac{900}{2500} = \frac{9}{25}.
\end{aligned}$$

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

Solution 2.20.5 All follows straightaway from the definitions in the main text. In the $G/G/1$ queue jobs arrive and depart in single units. Hence, $|A(n,t) - D(n,t)| \leq 1$, c.f., Eq. (2.30a). Then,

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

The left hand side goes to $\lambda\pi(n)$ as $t \to \infty$, and the right hand side to $\gamma\delta(n)$. Use the fact that we always assume, implicitly, that the system is stable, so that $\lambda = \gamma$. As a consequence $\delta(n) = \pi(n)$.

Solution 2.20.6 It took me some time to understand the details of part a. I then decided to read part b first, which clarified the intent of part a.

1.

```
>>> a = 3.
>>> b = 9.
>>> EA = (b+a)/2.
>>> EA
6.0
>>> labda = 1./EA # per minute
>>> labda
0.16666666666666666
>>> VA = (b-a)*(b-a)/12.
>>> CA2 = VA/(EA*EA)
>>> CA2
0.08333333333333333
>>>
>>> ES = 5.
```

```
>>> sigma = 2
>>> VS = sigma*sigma
>>> CS2 = VS/(ES*ES)
>>> CS2
0.16
>>>
>>> rho = labda*ES
>>> rho
0.8333333333333333
>>>
>>> Wq = (CA2+CS2)/2. * rho/(1.-rho) * ES
>>> Wq
3.041666666666665
```

2.

```
>>> Wq = (1.+CS2)/2.*rho/(1.-rho)*ES
>>> Wq
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.

Solution 2.20.7

1. Evident
2.

```
from math import sqrt

def MMc(labda, ES, c):
    rho = labda*ES/c
    return  rho**(sqrt(2*(c+1))-1)/(c*(1-rho)) * ES
```

As a test I first apply the code to the $M/M/1$ queue, for which we have closed form expression.

```
>>> labda = 5
>>> mu = 6
>>> ES = 1./mu
>>> rho = labda/mu
>>> print(MMc(labda, 1/mu,1), rho/(1-rho)*ES)
0.8333333333333328 0.8333333333333336
```

The results check. As a rule, you should always compare your results with known results. BTW, that is one of the reasons I prefer to code the formulas instead of using a calculator. Testing with code is relatively easy, whereas with a calculator it is impossible (You simply can't check what you typed at the calculator.)

Now for the case of the problem.

```
>>> labda = 5
>>> mu = 2
>>> ES = 1/mu
>>> c =3
```

```
>>>
>>> multi  = MMc(labda, ES, c) # multi server
>>> single  = MMc(labda, ES/c, 1) # single fast server
>>> print(multi, single)
0.7165109996820135 0.8333333333333328
```

Note that I use here that the multi-server formula reduces to the single-server formula by taking $c = 1$.

3.

```
>>> print(MMc(labda, ES, c)+ES, MMc(labda, ES/c, 1)+ES/c)
1.2165109996820136 0.9999999999999994
```

4. Lets first compute the waiting in queue for some values of c.

```
>>> for c in range(1,10):
...     print(MMc(labda, ES, c), MMc(labda, ES/c, 1))
...
-0.8333333333333334 -0.8333333333333334
-1.381879167974249 -1.25
0.7165109996820135 0.8333333333333328
0.12064648316732414 0.20833333333333334
0.03624611776407728 0.1
0.012956658155108965 0.059523809523809514
0.005061548428895368 0.03968253968253967
0.002092124098713698 0.028409090909090908
0.0009005311353996087 0.021367521367521368
```

Hmm, some results are plainly wrong; indeed, $c = 1$ is too small, and $\lambda/(c\mu) = \lambda/\mu = 5/2$. And this is larger than 1, and then the formula is not valid anymore. Only from $c \geq 3$ the results are OK.

BTW, I sometimes include the mistakes I make, just to show you the type of mistakes you have to take care to avoid.

```
>>> for c in range(3,10):
...     print(MMc(labda, ES, c)+ES, MMc(labda, ES/c, 1)+ES/c)
...
1.2165109996820136 0.9999999999999994
0.6206464831673242 0.33333333333333337
0.5362461177640773 0.2
0.5129566581551089 0.14285714285714285
0.5050615484288954 0.1111111111111111
0.5020921240987137 0.09090909090909091
0.5009005311353996 0.07692307692307693
```

This is interesting. The time in the $M/M/1$ system with the fast server becomes ever smaller, whereas the sojourn time in the $M/M/c$ converges to 1/2. The explanation is easy. With a fast server, the service time becomes small, whereas with many servers the time in queue becomes small, but the service time itself not.

The processing capacity of c servers that work at rate 1 is equal to one server that works at rate c. The, intuitive, difference between the $M/M/c$ queue and the $M/M/1$ queue with a fast server is that jobs spend shorter time in queue in the $M/M/c$ case—jobs can be

served in parallel, hence do not always have to wait in queue for service capacity—, but spend longer time in service—simply because the servers in the $M/M/c$ case works at rate 1, not at rate c.

5. To put things in perspective, see Figure 6.2.

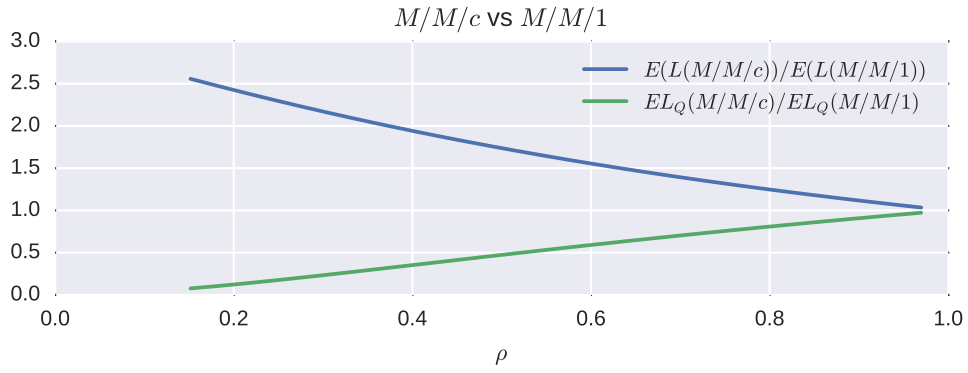


Figure 6.2: multi server versus single fast server; here the number of servers $c = 3$. We plot the ratio of the average number of jobs in the $M/M/3$ case and the $M/M/1$ queue with a fast server. (Note that by Little's law, the ratio between the waiting times is the same.) Clearly, jobs spend, on average, less time in queue in the $M/M/c$ case, but more time in total. This is as expected. In particular, if ρ is small, the time in the system must be dominated by the service time. Since the average service time is three times as short at the fast server, the total time in the system must be approximately three times as long in the $M/M/3$ case for ρ is small. On the other hand, when ρ is large, the queues are long, and the total time in the system is dominated by the queueing times. As queued jobs depart from the queue at rate 3, in both queueing systems, the average queueing times must be roughly equal. This we can also see clearly in the graph,

Here is the code I used to make the graphs.

```
from math import exp, factorial
import matplotlib.pylab as plt
import seaborn as sns
sns.set_context("paper")


def G(labda, mu, c):
    rho = labda / mu / c
    res = sum((c * rho)**n / factorial(n) for n in range(c))
    res += (c * rho)**c / ((1 - rho) * factorial(c))
    return res


def ELQ(labda, mu, c):
    rho = labda / mu / c
    g = G(labda, mu, c)
    return (c * rho)**c / (factorial(c) * g) * rho / (1 - rho)**2
```

```
def ELS(labda, mu, c):
    rho = labda / mu / c
    return rho * c


def EL(labda, mu, c):
    return ELS(labda, mu, c) + ELQ(labda, mu, c)


labda = 10
mu = 11

c = 3

R = range(5, c * mu)
Q = []
L = []
load = []
for labda in R:
    Q.append(ELQ(labda, mu, c) / ELQ(labda, c * mu, 1))
    L.append(EL(labda, mu, c) / EL(labda, c * mu, 1))
    load.append(labda / mu / c)

plt.figure(figsize=(6, 2))
plt.subplot(111)
plt.axis([0, 1, 0, 3])
plt.title("$M/M/c$ vs $M/M/1$")
plt.plot(load, L, label="$E(L(M/M/c))/E(L(M/M/1))$")
plt.plot(load, Q, label="$EL_Q(M/M/c)/EL_Q(M/M/1)$")
plt.xlabel("$\\rho$")
plt.legend()
plt.subplots_adjust(bottom=0.24)

# plt.show()
filename = "figures/multi_vs_single.pdf"
plt.savefig(filename)
plt.close()
```

The reader should observe that this code is not particularly efficient, many of the computations are done a few times. With recursion quite a number of shortcuts can be made.

Solution 2.21.1 We first consider the expectation. If it is given that $N = n$, then

$$\mathbb{E}[S_N \,|\, N = n] = \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \mathbb{E}[X_1] + \mathbb{E}[X_2] + \cdots \mathbb{E}[X_n] = n\,\mathbb{E}[X],$$

where the last equation follows from the fact that the X_i have the same distribution. With this,

$$\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{N} X_i \,\middle|\, N\right]\right] = \mathbb{E}[N\,\mathbb{E}[X]] = \mathbb{E}[X]\,\mathbb{E}[N].$$

This result in known as Wald's equation.

To compute the variance we use that $\mathbb{V}[S_N] = \mathbb{E}\left[S_N^2\right] - (\mathbb{E}[S_N])^2$. Thus, we first compute

$$\begin{aligned}\mathbb{E}\left[S_N^2 \,\middle|\, N=n\right] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} X_i\right)^2\right] = \mathbb{E}\left[\sum_{i=1}^{n} X_i^2 + \sum_{i\neq j}^{n} X_i X_j\right] \\ &= \sum_{i=1}^{n} \mathbb{E}\left[X_i^2\right] + \sum_{i\neq j} \mathbb{E}[X_i]\,\mathbb{E}\left[X_j\right],\end{aligned}$$

since $\mathbb{E}\left[X_i X_j\right] = \mathbb{E}[X_i]\,\mathbb{E}\left[X_j\right]$ by independence. Therefore, using that the $\{X_i\}$ are i.i.d. as X,

$$\mathbb{E}\left[S_N^2 \,\middle|\, N=n\right] = n\,\mathbb{E}\left[X^2\right] + n(n-1)(\mathbb{E}[X])^2.$$

With this,

$$\begin{aligned}\mathbb{E}\left[S_N^2\right] &= \mathbb{E}\left[\mathbb{E}\left[S_N^2 \,\middle|\, N=n\right]\right] \\ &= \mathbb{E}\left[N\,\mathbb{E}[X]^2 + N(N-1)(\mathbb{E}[X])^2\right] \\ &= \mathbb{E}[N]\,\mathbb{E}\left[X^2\right] + \mathbb{E}\left[N^2\right](\mathbb{E}[X])^2 - \mathbb{E}[N](\mathbb{E}[X])^2 \\ &= \mathbb{E}[N](\mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2) + \mathbb{E}\left[N^2\right](\mathbb{E}[X])^2 \\ &= \mathbb{E}[N]\,\mathbb{V}[X] + \mathbb{E}\left[N^2\right](\mathbb{E}[X])^2.\end{aligned}$$

And finally,

$$\begin{aligned}\mathbb{V}[S_N] &= \mathbb{E}\left[S_N^2\right] - (\mathbb{E}[S_N])^2 \\ &= \mathbb{E}[N]\,\mathbb{V}[X] + \mathbb{E}\left[N^2\right](\mathbb{E}[X])^2 - (\mathbb{E}[N])^2(\mathbb{E}[X])^2 \\ &= \mathbb{E}[N]\,\mathbb{V}[X] + \mathbb{V}[N](\mathbb{E}[X])^2.\end{aligned} \tag{6.5}$$

Solution 2.21.2

1. Only from below. If batches are very large, the fraction of time spent on setups becomes very small.

2. $$\lambda\,\mathbb{E}[S]$$

   Ensure you realized that you used Wald's rule.

3. The number of jobs that arrive, on average, during one production cycle must be smaller than the total amount of jobs that can be served, on average, during one cycle. The number of arrivals during the setup is $\lambda\mathbb{E}[S]$. The number of arrivals during serving the batch is $\lambda B/\mu$. Thus,
   $$\lambda(B/\mu + S) < B.$$

Solution 2.21.3 Yes. The availability is 93%. Since $\mathbb{E}[S_e] = \mathbb{E}[S]/A$, and if $\mathbb{E}[S]$ is known, $\mathbb{E}[S_e]$ follows.

Solution 2.21.4 No, from Zijm.Eq.1.51, the average repair time has to be known. The repair time here is of course the time it takes for a mechanic to show up at work again.

Solution 2.21.5 It might, but perhaps a normal distribution would be better. It makes sense to make a histogram of the recover times to see whether some clear pattern is present.

Besides this, I don't know how sensitive the result for C_e^2 is on the distribution of the repair times. Perhaps it is not that sensitive, so in that case it would be ok to simply use the exponential distribution.

The sensitivity study would be an interesting topic for simulation.

Solution 2.21.6 We need to make an assumption about the distribution of the repair times. Inferring from the text, the repair time is always 2 days. Lets also assume that all jobs accumulate in front of the broken machine, in other words, the broken machine is part of the job routing of each job. Then

$$\mathbb{P}\{N(2) > 20\} = \sum_{n=21}^{\infty} e^{-5\cdot 2}\frac{10^n}{n!}.1 - \sum_{n=0}^{20} e^{-5\cdot 2}\frac{10^n}{n!} = 1 - 0.9984,$$

i.e., very small. This is the code I used:

```
>>> from math import exp, factorial
>>> exp(-10) * sum((10)**n / factorial(n) for n in range(21))
0.9984117393381421
```

Solution 2.21.7 If the shop already contains 10 jobs, somewhere upstream of the broken machine, then

$$\mathbb{P}\{N(2) > 10\} = 1 - 0.58.$$

where I used

```
>>> exp(-10) * sum((10)**n / factorial(n) for n in range(11))
0.5830397501929855
```

Solution 2.21.8 Cleaning times will be pretty constant. Changing dies, or other machine parts, is also typically quite predictable, although it can take a lot of time, in particular in case a crane or other heavy machinery is needed to replace parts. If the machine require temporary adjustments, then the variation in setup times may be quite a bit higher.

Solution 2.21.9 Then the effective service times, and in particular, C_e^2 will be quite a bit bigger. It is preferable to avoid such a situation.

Mathematically, it is only given that N_s is a random variable. As, however, this does not state anything about its distribution, we cannot make any general claim. The intent of the problem is to have you check the relevant formulas and notice that the variance of N_s appears in the formulas.

Solution 3.1.1 The processing rates are $\mu_1 = 1/1$ and $\mu_2 = 1/2$ per hour, respectively. The utilizations $\rho_i = \lambda_i/\mu_i$ are 1/3 and 2/3, respectively. Thus, machine 2 is the bottleneck, as it has the highest utilization.

Since these machines are not related by a routing (jobs going from one machine to another), computing T_0 and so on does not make much sense for the network. We can just consider each machine on its own. Then $W_0 = 2$—we need two jobs to keep the machines busy.

Solution 3.1.2 The raw processing time is just the sum of the processing times at each station. All stations have the same arrival rate, hence machine, being the slowest, is the bottleneck. Hence,

$$\begin{aligned} T_0 &= 2+3+2 = 7 \\ r_b &= \mu_2 = \frac{1}{3} \\ W_0 &= r_b T_0 = \frac{7}{3}. \end{aligned}$$

Solution 3.1.3 Realize that machines at a station are in parallel, not in series. Thus, when we add capacity to station 2, we don't add an extra processing step to station 2, we increase capacity.

Now

$$r_2 = \frac{1}{3} + \frac{1}{3} = \frac{2}{3},$$

so that station 2 is no longer the bottleneck. Station 1 and station 3 are the bottlenecks.

$$\begin{aligned} r_b &= \frac{1}{2} \\ T_0 &= 2+3+2 = 7 \\ W_0 &= r_b T_0 = \frac{7}{2}. \end{aligned}$$

Often, students think that the processing rate is the inverse of the processing time, i.e., $t = r^{-1}$. This is not the case for multi-server stations. Thus, realize that the raw processing time at station 2 is still 3 hours, not 3/2.

Solution 3.1.4 Now,

$$r_2 = \frac{1}{3} + \frac{1}{4} = \frac{7}{12} > \frac{1}{2},$$

hence, station 1 and 3 are bottlenecks.

However, t_2 cannot be 3 hours anymore: some jobs will spend 4 hours at station 2 rather than 3 hours. In fact, the computation of the raw processing time at station 2, t_2 say, is a bit tricky. Some students reason that both machines serve half of the jobs, so that $T_2 = (3+4)/2$ hours. This, however, is not quite realistic. Like this, you would load the slow machine all of the time, and the faster machine would be idle quite a bit of the time.

Rather than spreading the jobs evenly over the two machines, we can also assume that each machine takes in work in propertion to its processing rate. Then we reason like this. Consider 12 hours. In those 12 hours, the slow machine processes 3 jobs with duration 4 and the fast machine processes 4 jobs with duration 3. Therefore:

$$t_2 = \frac{3}{7}4 + \frac{4}{7}3 = \frac{24}{7}.$$

Another way to get this is like this. We can 'stuff' station 2 with jobs. The most jobs that fit in simultaneously is 2, one job per machine. Then, with Little's law,

$$t_2 = \frac{W}{r_2} = \frac{2}{7/12} = \frac{24}{7},$$

i.e., the same as the other answer.

Now, finally,

$$\begin{aligned} r_b &= \frac{1}{2} \\ T_0 &= 2 + \frac{24}{7} + 2 = 4 + 3 + \frac{3}{7} = 7\frac{3}{7} \\ W_0 &= r_b T_0. \end{aligned}$$

Finally, and the most realistic, is that the fast machine is always working, and the slow machine covers the rest. In that case, suppose that the fast machine processes jobs at rate r_1 and the slow at rate r_2. Then, since jobs arrive with rate r_a, the fast machine processes a fraction of r_1/r_a of the jobs, and the slow machine processes the left overs, i.e., a fraction of $(r_a - r_1)/ra$ of the jobs. The raw processing time then becomes

$$\begin{aligned} t_2 &= \frac{r_1}{r_a} t_1 + \frac{r_a - r_1}{r_a} t_2 \\ &= \frac{1/3}{1/2} 3 + \frac{1/2 - 1/3}{1/2} 4 \\ &= 2 + \frac{1/6}{1/2} 4 \\ &= 2 + \frac{4}{3} = 3\frac{1}{3}, \end{aligned}$$

and then,

$$T_0 = 2 + t_2 + 2 = 7\frac{1}{3},$$

and this is a tiny bit less then the previous way of organizing things.

Solution 3.1.5 Because T_0 increases and/or the bottleneck rate increases. In both cases, we need more W_0 to fill the network and achieve that the throughput can be equal to r_b in the best case.

Solution 3.1.6 Now the slow machine at station 2 becomes superfluous. The fast machine at station 2 can cope with all jobs that arrive from station 1.

$$\begin{aligned} r_b &= \frac{1}{2} \\ T_0 &= 2 + 1 + 2 = 5 \\ W_0 &= r_b T_0. \end{aligned}$$

Thus, why would you assign part of the jobs to the slow machine at station 2? There is no queue, the fast machine can cope with all the jobs. Moreover, assigning any job to the slow machines just adds to the cycle time.

Taking $T_2 = (1 + 3)/2 = 2$ is plainly silly.

Solution 3.1.7 See above. I would not remove the slow machine. In real life there is variability: the fast machine might fail for instance. Spare capacity is useful. However, if it is very costly to keep the slow machine operational, I would scrap it.

Solution 3.1.8

$$r_2 = \frac{1}{3} + \frac{1}{7} = \frac{10}{21} < \frac{1}{2}$$

hence station 2 is still the bottleneck

$$\begin{aligned} T_2 &= \frac{3}{10}7 + \frac{7}{10}3 = \frac{42}{10} = \frac{21}{5}\text{hour} \\ T_0 &= 2 + \frac{21}{5} + 2 \\ r_b &= \frac{10}{21} \\ W_0 &= r_b T_0. \end{aligned}$$

Note that we add capacity, so that we can process more jobs per unit time, but the raw processing time increases!

Solution 3.1.9 Now there are two loops. One third of the jobs pass all three stations. Station 1 can process these job at rate

$$\frac{1}{3}r_1 = \frac{1}{3}\frac{1}{2} = \frac{1}{6}$$

per hour. Since this is smaller than r_2 and r_3, station 1 is the bottleneck rate for this loop. Thus, W_0 in this loop is $r_b T_0 = 1/6 \cdot 7 = 7/6$.

The other loop only contains station 1. The bottleneck rate in this loop is

$$r_1\frac{2}{3} = \frac{1}{2}\frac{2}{3} = \frac{1}{3}.$$

Thus, $W_0 = r_b T_0 = 1/3 \cdot 2 = 2/3$ for this loop.

Hence, the total wip is $7/6 + 2/3 = 11/6$ and

$$T_0 = \frac{1}{3}7 + \frac{2}{3}2 = \frac{11}{3},$$

since 1/3 take the long loop with $T = 7$ and 2/3 the short loop with $T = 2$. Thus, T_0 is the weighted average raw processing time.

Another way to analyze this situation is like this. Station 1 is the bottleneck, because its utilization is 1, while

$$\begin{aligned} \rho_2 &= \lambda_2 t_2 = \frac{1}{2}\frac{1}{3}3 = \frac{1}{2} \\ \rho_3 &= \lambda_3 t_3 = \frac{1}{2}\frac{1}{3}2 = \frac{1}{3}. \end{aligned}$$

Since station 1 is the bottleneck in this network, $r_b = 1/2$. Since $T_0 = 11/3$, which we obtain as the weigthed average computation above, we must have that

$$W_0 = r_b T_0 = \frac{1}{2}\frac{11}{3} = \frac{11}{6},$$

which agrees with our earlier result.

Solution 3.1.10 Now stations 1 and 3 are bottlenecks with $r_b = 1/2$. For T_0 we have

$$T_0 = \frac{1}{5}7 + \frac{4}{5}4 = \frac{23}{5}.$$

Thus,

$$W_0 = r_b T_0 = \frac{1}{2}\frac{23}{5} = \frac{23}{10}.$$

Solution 3.1.11 As indicated in the problem, I don't have a solution for this problem.

Solution 3.2.1

1. The output rate is λ if $\lambda \le \mu$.
2. Because jobs do not leave at rate μ.
3. The simplest, but reasonable, inter-departure density would be $\lambda e^{-\lambda t}$. So this is what we will try to prove.
4. $\mathbb{P}\{I\} = p_0 = \pi_0$. Recall that π_0 is what the jobs see upon arrival and $p(0)$ is the time-average. By PASTA these are the same. Recall also that $p_0 = 1 - \rho$.
5. When job n finds an empty system, job $n-1$ left an empty system behind. Thus, we first have to wait for an inter-arrival time X_n. Then, since job n's service starts right away, he leaves when $D_n = D_{n-1} + X_n + S_n$.
6. By the previous point, the density of $D_n - D_{n-1}$ is the same as the density of $X_n + S_n$. Therefore, by conditioning,

$$\begin{aligned} f_{X+S}(t) &= \mathbb{P}\{X + S \in \mathrm{d}t\} \\ &= \int \mathbb{P}\{S + x \in \mathrm{d}t\}\, \mathbb{P}\{X \in \mathrm{d}x\} \\ &= \int_0^t f_S(t-x) f_X(x)\, \mathrm{d}x \\ &= \int_0^t \mu e^{-\mu(t-x)} \lambda e^{-\lambda x}\, \mathrm{d}x \\ &= \lambda\mu e^{-\mu t} \int_0^t e^{x(\mu-\lambda)}\, \mathrm{d}x \\ &\frac{\lambda\mu}{\lambda-\mu}(e^{-\mu t} - e^{-\lambda t}). \end{aligned}$$

7. $\mathbb{P}\{B\} = \mathbb{P}\{\text{busy}\} = \rho$.

8.

$$\begin{aligned}
f_D(t) &= f_{X+S}(t)\mathbb{P}\{I\} + f_S\,\mathbb{P}\{B\} \\
&= (1-\rho) f_{X+S}(t) + \rho\mu e^{-\mu t} \\
&= (1-\rho)\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \rho\mu e^{-\mu t} \\
&= (1-\frac{\lambda}{\mu})\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \rho\mu e^{-\mu t} \\
&= \frac{\mu-\lambda}{\mu}\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \frac{\lambda}{\mu}\mu e^{-\mu t} \\
&= \frac{\mu-\lambda}{\mu}\frac{\mu\lambda}{\lambda-\mu}\left(e^{-\mu t} - e^{-\lambda t}\right) + \lambda e^{-\mu t} \\
&= -\lambda\left(e^{-\mu t} - e^{-\lambda t}\right) + \lambda e^{-\mu t} \\
&= \lambda e^{-\lambda t}.
\end{aligned}$$

9. If the server busy when a job arrives is the same as the density of the inter-departure time, where we also use the memoryless property of the exponential distribution. Therefore, the density is $\mu e^{-\mu t}$.

10. Burke's law gives that the inter-departure times are independent. The implication of all this is that the inter-departures times form a set of i.i.d. exponentially distributed random variables. Hence, the departure process is a Poisson(λ) process.

Solution 3.2.2 Because jobs cannot leave faster than they arrive.

Solution 3.2.3 . Observe from Question 1 that the inter-departure times of the $M/M/1$ queue are also independent and identically exponentially distributed with rate λ. Since the arrival process at the second station is the departure process of the first station, it must be that arrival process at the second station is also Poisson with rate λ. Interestingly, from the perspective of the second station it is as if there is not first station.

Solution 3.2.4 The question is not well specified. We know from Burke's law that the arrival *process* at the second station is Poisson. If, however, we know that station 1 is empty, then it is unlikely that a job will arrive at station 2 in the very near future.

Note that only the steady-state distributions of the queue lengths are independent. Once you have information about the state of one of the queues, then certainly this is not in 'steady-state'.

Solution 3.2.5 Simple algebra. (I am not going to write it out here. If you are willing to provide me the answer in latex I'll include it.)

Solution 3.2.6 Linear algebra is quite useful here!

Observe that P_{ij} is the probability that a job, after completing its service at node i, moves to node j. Then $\sum_{j=1}^{M} P_{ij}$ is the probability that a job moves from node i to another node in the network, i.e., stays in the network, while P_{i0} is the probability that a job departing from node i leaves the network, in other words, the job is finished. When $\sum_{j=1}^{M} P_{ij} < 1$, then more jobs enter node i from the network than that node i sends 'back' into the network. Conceptually, node i 'leaks jobs'.

Now, consider some node k such that $P_{ki} > 0$, then the probability that a job that starts at node k, moves to node i and then leaves the network is equal to $P_{ki}Pi0$. Thus, since $P_{ki} > 0$ and $P_{i0} > 0$, the probability that a job leaves the network from note k in two steps is positive. More specifically, $P^2_{k0} = \sum_{j=0}^{M} P_{kj}P_{j0} \geq P_{ki}P_{i0} > 0$.

The irreducibility assumption implies that in at most M steps it is possible to reach, with positive probability, any node from any other node in the network. Thus, for any node j to any other node k there is a sequence of nodes $j_1, j_2, \ldots j_{M-1}$ such that $P^M_{jk} \geq P_{jj_1}P_{j_1j_2}\cdots P_{j_{M-1}k} > 0$.

Thus, if there is a node i such that $P_{i0} > 0$, then it is possible from any node that sends jobs to node i directly to leave the network in two steps. Likewise, when node i can be reached from node k in n steps, say, then $P^{n+1}_{k0} \geq P^n_{ki}P_{i0} > 0$, i.e., in at most $n+1$ steps it is possible to leave the network from such node k. This implies, in particular, that for all nodes $k = 1,2,\ldots M$, i.e., all nodes in the network, $P^{M+1}_{k0} > 0$. For this reason we consider P^{M+1} in the hint.

As a final remark for students with knowledge of Markov chains, observe that the routing matrix P does not correspond to the transition matrix of a recurrent Markov chain. Since for at least one row i, $\sum_{j=1}^{N} P_{ij} < 1$, the matrix P is sub-stochastic. Hence, a Markov chain induced by P cannot be irreducible, because for this to happen, the chain must stay in some absorbing set with probability 1.

Solution 3.2.7 Since M is finite, and $k \leq M$, the set of numbers P^{M+1}_{k0} is finite. This, together with the fact that $P^{M+1}_{k0} > 0$ for all k, implies that there is some number $\epsilon > 0$ such that $P^{M+1}_{k0} > \epsilon$. Hence, for all entries $k = 1,2,\ldots M$, we have that $P^{M+1}_{kj} < 1-\epsilon$. This, in turn, implies that $P^{2(M+1)}_{kj} < (1-\epsilon)^2$, and so on, so that for any n, $P^{n(M+1)}_{kj} < (1-\epsilon)^n$. This implies, in more general terms, that the entries of P^n decrease at some geometric rate to 0.

It is well known that for any bounded sequence x_i and $0 \leq \alpha < 1$, $\sum_{i=0}^{\infty} x_i\alpha^i < \infty$. Applying this insight to the entries of P^n it follows that $\sum_{n=0}^{\infty} P^n_{jk} < \infty$.

Finally, applying $\lambda = \gamma + \lambda P$ recursively, we get

$$\lambda = \gamma + \lambda P = \gamma + (\gamma + \lambda P)P = \gamma(1+P) + \lambda P^2 = \gamma(1+P+P^2) + \lambda P^3 \to \gamma \sum_{n=0}^{\infty} P^n.$$

By the above reasoning this last sum is well defined, and finite. (By the way, the above argument is not necessarily valid for matrices P that are infinite, since then $\inf\{P^M_{ik}\}$ need not be strictly positive.)

Another interesting way to see all this is by making the simplifying assumption that P is a diagonalizable matrix. (The argument can be generalized to include matrices reduced to Jordan normal form, but this gives optimal clutter, but does change the line of reasoning in any fundamental way.) In that case, there exists an invertible matrix V and a diagonal matrix Λ with the eigenvalues λ_i of P on its diagonal such that

$$P = V^{-1}\Lambda V.$$

But then $P^2 = V^{-1}\Lambda V \cdot V^{-1}\Lambda V = V^{-1}\Lambda^2 V$, and in general $P^n = V^{-1}\Lambda^n V$. If each eigenvalue λ_i is such that its modulus $|\lambda_i| < 1$, then $\Lambda^n \to 0$ geometrically fast, hence $P^n \to n$ geometrically fast, hence the sequence of partial sums $\sum_{n=1}^{N} P^n$ must converge to some finite number.

So, we are left with proving that the eigenvalues of P must have modulus less than 1. This fact follows from Gerschgorin's disk theorem, which I include for the interested student. Define the disk $B(a,r) = \{z \in \mathbf{C} || z-a| \leq r\}$, i.e., the set of complex numbers such that the distance to the center $a \in \mathbf{C}$ is less than or equal to the radius r. With this, the Gerschgorin disks of a matrix

are defined as $B(a_{ii}, \sum_{j\neq i}|a_{ij}|)$, i.e., the disk with center a_{ii}, the diagonal element of A on the ith row, and radius $\sum_{j\neq i}|a_{ij}|$ of the modulus of the elements of A on row i not an the diagonal. Then the theorem says that all eigenvalues of A lie in the union of Gerschgorin disks, i.e., all eigenvalues $\lambda_i \in \bigcup_i B_{a_{ii},\sum_{j\neq i}|a_{ij}}$).

Assume for notational simplicity that for each row i of P we have that $\sum_j a_{ij} < 1$. (Otherwise apply the argument to P^{M+1}.) Then this implies for all i that

$$a_{ii} + \sum_{j\neq i} a_{ij} < 1.$$

Since all elements of P are non-negative, this also implies that

$$-1 < a_{ii} - \sum_{j\neq i} a_{ij} \leq a_{ii} + \sum_{j\neq i} a_{ij} < 1.$$

With this and using that a_{ii} is a real number, hence lies on the real number axis, it follows that the modulus of all elements of the disk $B(a_{ii}, \sum_{j\neq i} a_{ij})$ is smaller than 1. As this applies to any row i, all disks lie strictly within the complex unit circle. But then, by Gerschgorin's theorem, all eigenvalues of P also lie strictly in the unit circle, hence all eigenvalues have modulus smaller than 1.

Solution 3.2.8 Take $n_i < c_i$. Then $\Pi_{k=1}^{n_i} \min\{k, c_i\} = \Pi_{k=1}^{n_i} k = n_i!$, and $(c_i\rho_i)^{n_i} = (\lambda_i/\mu_i)^{n_i}$. If $n_i \geq c_i$, then $\Pi_{k=1}^{n_i} \min\{k, c_i\} = c_i! c_i^{n_i - c_i}$, and $(c_i\rho_i)^{n_i} = (\lambda_i/\mu_i)^{n_i} c_i^{n_i}$..

Solution 3.2.9

$$P = \begin{pmatrix} \alpha & 1-\alpha \\ \beta_1 & \beta_2 \end{pmatrix}.$$

$$(\lambda_1, \lambda_2) = (\gamma, 0) + (\lambda_1, \lambda_2)P.$$

Solving first for λ_2 leads to $\lambda_2 = (1-\alpha)\lambda_1 + \beta_2\lambda_2$, so that

$$\lambda_2 = \frac{1-\alpha}{1-\beta_2}\lambda_1.$$

Next, using this and that $\lambda_1 = \alpha\lambda_1 + \beta_1\lambda_2 + \gamma$ gives with a bit of algebra

$$\begin{aligned} \gamma &= \lambda_1(1-\alpha) - \beta_1\lambda_2 \\ &= \lambda_1\left(1 - \alpha - \beta_1\frac{1-\alpha}{1-\beta_2}\right) \\ &= \lambda_1(1-\alpha)\left(1 - \frac{\beta_1}{1-\beta_2}\right) \\ &= \lambda_1(1-\alpha)\frac{1-\beta_1-\beta_2}{1-\beta_2}. \end{aligned}$$

Hence,

$$\lambda_1 = \frac{\gamma}{1-\alpha}\frac{1-\beta_2}{1-\beta_1-\beta_2}.$$

Thus,

$$\lambda_2 = \frac{1-\alpha}{1-\beta_2}\lambda_1 = \frac{\gamma}{1-\beta_1-\beta_2}.$$

We want of course that $\lambda_1 < \mu_1$ and $\lambda_2 < \mu_2$. With the above expressions this leads to conditions on α, β_1 and β_2. Note that we have three parameters, and two equations; there is not a single condition from which the stability can be guaranteed.

If $\alpha \uparrow 1$, the arrival rate at node 1 explodes. If $\beta_1 = 0$ no jobs are sent from node 2 to node 1.

Solution 3.2.10 Yes, the network remains a Jackson network. By Burke's law the departure process of each node is Poisson. In one of the earlier questions we derived that splitting (also known as thinning) and merging Poisson streams again lead to Poisson streams. The departures from node j to node k forms a thinned Poisson stream. The external arrivals plus internal arrivals are merged into one Poisson stream, hence the arrivals at a station also form a Poisson stream.

Observe that the exponentiality of the service times and external inter-arrival times and Burke's law are essential for the argument.

Solution 3.3.1 The answer to this problem is of course not clear cut. A well balanced answer requires some extra information, such as which of the two stations has the larger variability. However, we can provide some general insights into this problem.

We first compute the expected waiting time in the queues at both stations for a reference situation. Then we derive similar expressions for the waiting times in case we spend all money on the second or first station.

The reference situation As a reference scenario, suppose that the arrival process has a Poisson distribution with rate λ, and that the service distributions at the first and second server, respectively, are exponential with rate μ_1 and μ_2, respectively. This is, from an analytic point of view, about the simplest situation we can consider. We now focus on the waiting times for each station separately.

The first queue is the familiar $M/M/1$ queue. (Why?) For the sequel it is important to observe that the departure process of the first station, i.e., the distribution of times between jobs leaving the first station, is the same as the arrival process. Consequently, the inter-departure times are also exponentially distributed with parameter λ. (However, the service times are exponentially distributed with parameter μ_1.) This claim can be proved, but is not immediate.

What can we say about the second station? Clearly, the jobs departing at the first stations are the arrivals at the second station. Hence, this departure process being exponential with rate λ, the inter-arrival times at the second station are also exponential with rate λ. Consequently, the second queue is also an $M/M/1$ queue.

It is well known that, for this case, the total waiting time in the system, i.e., the time spent in both queues, is the sum of the waiting times at the first and second station. As each is an $M/M/1$ queue, the total waiting time has the form:

$$\begin{aligned} E(W) &= E(W_1) + E(W_2) \\ &= \frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2}, \end{aligned} \tag{6.6}$$

where $\rho_i = \lambda/\mu_i$ and $E(S_i) = 1/\mu_i$, for $i = 1,2$.

Case1: Spend all the money on the second server Now we might spend the money on the second machine. How would this influence the total waiting time $E(W)$?

As we reduce the variability of the second server, the service process is no longer exponential. However, the arrival process at the second station is still Poisson. As a consequence, the queueing discipline changes to the $M/G/1$ queue. The expected waiting time for this case has

the form:

$$E(W_2) = \frac{1 + C_{s,2}^2}{2} \frac{\rho_2}{1-\rho_2} \frac{1}{\mu_2}, \tag{6.7}$$

where $C_{s,2}$ is the coefficient of variation of the service process of the second server.

Suppose the money we have suffices to entirely remove the variability of the second server. (This is of course also the best we can possibly achieve.) This yields that the coefficient of variation $C_{v,2} = 0$. As an aside, observe that as all variability has been removed, the service process is entirely deterministic: each service takes exactly the same amount of time. As a result, we arrive at the $M/D/1$ queue to model the queueing process at the second station.

The expected waiting time for the $M/D/1$ queue follows immediately from (6.7) by setting $C_{v,2} = 0$:

$$E(W_2) = \frac{1}{2} \frac{\rho_2}{1-\rho_2} \frac{1}{\mu_2}. \tag{6.8}$$

Clearly, this is half the waiting time of the $M/M/1$ queue.

Since we do not change the first station in any way, this is still an $M/M/1$ queue. Thus, the total waiting time for this scenario becomes:

$$E(W) = \frac{\rho_1}{1-\rho_1} \frac{1}{\mu_1} + \frac{1}{2} \frac{\rho_2}{1-\rho_2} \frac{1}{\mu_2}.$$

Case 2: Spend the money on the first server Analogous to the previous situation, suppose we can set the coefficient of variation $C_{1,v}$ of the first server to zero. Thus, this becomes an $M/D/1$ queue, so that, similar to (6.8):

$$E(W_1) = \frac{1}{2} \frac{\rho_1}{1-\rho_1} \frac{1}{\mu_1}.$$

Contrary to the $M/M/1$ queue, the inter-departures of the $M/D/1$ queue are not exponentially distributed. Rather, as long as the first server is busy, they are deterministic, i.e., equally separated in time. Only when the first server is idle—these idle times still have exponential distribution—there are no regular departures. Thus the departure process is not deterministic. However, for the sake of simplicity, we assume in the sequel that the departure process *is* deterministic.

As we previously remarked, the departure process of the first station forms the arrival process at the second station. Since the departures are assumed to be deterministic, the arrivals at the second station are also deterministic. The service times, however, are still exponential. (Recall we do not spend money on the second server to reduce its variability). Thus, the second station can be modeled as the $D/M/1$ queue. For this queue we need to derive an expression for the waiting time. The simplest approximation is follows from an expression for the waiting time of the $G/G/1$ queue.

It is well known that the expected waiting time for the the $G/G/1$ queue has the approximate form:

$$E(W_2) = \frac{C_{a,2}^2 + C_{s,2}^2}{2} \frac{\rho_2}{1-\rho_2} \frac{1}{\mu_2}, \tag{6.9}$$

Clearly, for our case, the coefficient of variation $C_{a,s}$ of the arrival process becomes, approximately, 0, while $C_{s,2} = 1$, since the service process is still exponential. Hence,

$$E(W_2) = \frac{1}{2} \frac{\rho_2}{1-\rho_2} \frac{1}{\mu_2}. \tag{6.10}$$

Combining (6.9) and (6.10), the total waiting time becomes:

$$E(W) = \frac{1}{2}\frac{\rho_1}{1-\rho_1}\frac{1}{\mu_1} + \frac{1}{2}\frac{\rho_2}{1-\rho_2}\frac{1}{\mu_2},$$

which is half the waiting time of the reference scenario, and still less than the first scenario.

Conclusion As a general guide line, it seems best to reduce the variability at the first station. The main point to remember is that reducing the variability of the service process at the first station also reduces the variability of the arrival processes at the second station. Thus, we gain at two locations of the chain of stations, rather than at one.

Interestingly, the departure process of the $D/M/1$ queue is not Poisson; the coefficient of variation is less than 1. Hence, if there were a third station, also its input process would behave better than Poisson. Consequently, reducing variability upstream reduces (in general) the variability of the arrival processes at all stations downstream, reducing the waiting time all along the chain.

Solution 3.3.2

1. This is really a neat problem. Please spend serious time on it to solve before looking at the answer. It requires some ingenuity on your part.

   Let p_{ij} be the fraction of time that the system contains j taxi cabs and i riders. I assume that all members of a party of riders can be served by a single cab (that is, the parties do not excees the capacity of a cab and all members of a party have the same destination). For clarity, write μ for the rate at which cabs arrive, and λ for the arrival rate of parties of riders. Then:

$$\begin{aligned}
\lambda p_{0,3} &= \mu p_{0,2} \\
(\lambda+\mu)p_{0,2} &= \mu p_{0,1} + \lambda p_{0,3} \\
(\lambda+\mu)p_{0,1} &= \mu p_{0,0} + \lambda p_{0,2} \\
(\lambda+\mu)p_{0,0} &= \mu p_{1,0} + \lambda p_{0,1} \\
(\lambda+\mu)p_{1,0} &= \mu p_{2,0} + \lambda p_{0,0} \\
(\lambda+\mu)p_{2,0} &= \mu p_{3,0} + \lambda p_{1,0}
\end{aligned}$$

   And so on. Thus, it is left to compute p_{ij}. Observe that the scheme of the equations is precisely the same as the scheme of the M/M/1 queue, just the number of the states is different. Let q be the number of jobs in an M/M/1 queue. Some thought will reveal that the queueing system with cabs and parties can be mapped to an equivalent M/M/1 queueing system. In fact, consider the following table

j	i	q
3	0	0
2	0	1
1	0	2
0	0	3
0	1	4
0	2	5

and so on. Therefore, in general, it must be that

$$q = 3 - j + i.$$

From the M/M/1 queue we know right away that $p_q = \rho^q(1-\rho)$. With the above relation we can therefore immediately find that $p_{ij} = \rho^{3-j+i}(1-\rho)$, save that i and j must satisfy the constraints imposed by the model.

2. The expected number of cabs waiting must be

$$1p_{0,1} + 2p_{0,2} + 3p_{0,3}$$

and the expected number of parties waiting must be $\sum_{j=1}^{\infty} jp_{j,0}$

```
labda = 12.  # per hour
mu = 15.  # per hour
rho = labda/mu

def p(i,j):
    q  = 3 - j + i
    return rho**q*(1.-rho)
```

Expected number of cabs waiting:

```
>>> Lc = sum(j*p(0,j) for j in range(0,4)) # recall this sums up to 4,
not including 4
>>> Lc
1.0479999999999998
```

To compute the expected number of parties waiting we formally have to sum to infinity. Rather than doing the algebra, I chose to truncate the summation at an i such that $\rho^i \ll 1$, i.e., negligble. Truncating at 30 seems reasonable enough:

```
>>> trunc = 30
>>> rho**trunc
0.0012379400392853823
```

Hm. At second thought this is not yet really small.

```
>>> trunc = 50
>>> rho**trunc
1.4272476927059638e-05
```

This is better. Now go for what we want to know:

```
>>> Lp = sum(i*p(i,0) for i in range(trunc))
>>> Lp
2.0476053945579213
```

3. This is tricky. I first, naively, computed $W_q = L_c/\mu$. This seems to make sense, as cabs arrive at rate μ, so that this expression follows from a standard application of Little's law. However, this is wrong, of course. When using Little's law to relate the number of jobs in queue (i.e., in the M/M/1 queue) and the queueing time we need to use λ, not μ. Similarly (and more formally by the mapping developed in part a), for our cab system we also need to use λ.

```
>>> Wq = Lc/labda
>>> Wq
0.08733333333333332
```

Thinking in templates is often useful, but makes one sloppy...

What would be the impact of allowing 4 cabs? Funny question, and with the above, trivial to answer.

```
def p(i,j):
    q  = 4 - j + i
    return rho**q*(1.-rho)


>>> Lc = sum(j*p(0,j) for j in range(0,4))
>>> Lc
0.8383999999999999
>>>
>>> Lp = sum(i*p(i,0) for i in range(trunc))
>>> Lp
1.638084315646337
```

Solution 3.4.1

1. The routing matrix P is M times M, hence contains M^2 parameters, but often most of them will be zero.
2. λ_{0i} is the arrival rate of external jobs to station i, hence M parameters.
3. C_{0i}^2 is the SCV of the interarrival times at station i, hence M parameters.
4. C_{si}^2 is the SCV of the service times at station i, hence M parameters.
5. The expected service times $\mathbb{E}[S]_i$ at station i, hence M parameters.
6. The number of servers c_i at station i, hence M parameters.

Solution 3.4.2 From the $G/G/1$ waiting time formula, it is obvious that by reducing the variability of the service times the average waiting time will also reduce. The results to be developed in this section will show that when the service times become less variable, the inter departure times also become less variable. As a a consequence, the downstream machine sees a more regular arrival process, hence, by the $G/G/1$ waiting time, will have less average waiting time. But also its departure process will go down, so that the waiting time at the next station also decreases, and so on. Thus, the entire chain benefits from reducing variability of processing times at the first station.

However, all depends of course on the cost. If it is very expensive to reduce variability at the first station, or if this station is already very 'regular', it might be better to improve the second station. Thus, as always, what to do depends on the context, financially and technically.

Solution 3.4.3 If it would be cheap. However, as the previous question should show you, by reducing variability at the first station the waiting time of all stations along the line goes down. Thus, by improving the first or second machine the total average waiting time may go down more than by only improving the situation at the last, bottleneck, machine.

Thus, we need models, simulations and/or experiments to find out what is the best option.

Solution 4.1.1 The routing matrix P is

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

Solving $V = VP$ leads to $(V_0, V_1, V_2) = (V_2/2, V_0 + V_2/2, V_1)$. Thus, from the last item, $V_2 = V_1$, and from the first $V_0 = V_2/2$. Since $V_0 = 1$, it follows that $V_2 = 2V_0$ and $V_1 = V_2 = 2V_0$.

Solution 4.1.2 TBD.

Solution 4.1.3 If a part would need refitting or repositioning at the load/unload station, the part would visit the load/unload station more than once during its stay in the network. The visit ratio of the load/unload station can then no longer be set to 1.

Solution 4.1.4 Lets number the states from 1 to 4. If station 1 feeds into station 2, and so on, and station 4 into station 1, then

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

The visit ratios are, of course, $V_1 = V_2 = V_3 = V_4$.

Solution 4.1.5 We number station a as 1, and so on.

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1/5 & 0 & 4/5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 2/3 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Since, by definition, $V_1 = 1$, it suffices to express the other visit ratios in terms of V_1. From elementary linear algebra, the second column implies that $V_1 = V_2$, and the third that $4/5V_2 = V_3$, so that $V_3 = 4/5V_1$. From the fourth column $1/3V_3 = V_4$, thus, $V_4 = 1/3 \cdot 4/5V_1 = 4/15V_1$. From the fifth column $2/3V_4 = V_5$, hence $V_5 = 8/45V_1$. Finally, from the sixth column, $V_4 + V_5 = V_6$, hence $V_6 = (4/15 + 8/45)V_1$.

For later courses on Markov chains, it is important to note the following. Write the visit ratio equation $V = VP$ as $V(1 - P) = 0$ where 1 is the indicator matrix. Clearly, V is must be a left

eigenvector of the matrix $1-P$ with eigenvalue 0 (recall that $V(1-P) = 0 = 0 \cdot V$). Thus, at least one row or column of the matrix $1-P$ is superflous to solve for V.

Solution 4.1.6 Except for the normalization constant the expressions are the same, c.f., Eq.1.36 and Eq. 1.37 of Zijms book, and Question 8.

Solution 4.1.7 Consider network one with routing $1 \to 3 \to 4 \to 2 \to 1$, each transition occuring with probability one, i.e.,

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

All nodes are is visited equally often. Another network could correspond to the routing $1 \to 2 \to 3 \to 4 \to 1$. Yet another would be this:

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

Observe that I choose the rate such that each node receives the same fraction of traffic.

For the interested: There must be many such 'equivalent' networks, but a general method to classify all equivalent networks seems hard. The question comes down to finding all stochastic matrices P that have at least one (left) eigenvector V in common. Observe that the visit ratio equation $VP = P$ implies that V is a left eigenvector with eigenvalue 1.

Solution 4.1.8 Since $M = 2$, we have three stations: Stations 0, 1, and 2. We also have $N = 2$ jobs. Thus, the states are $(2,0,0)$ (meaning that the load/unload station has 2 jobs, and stations 1 and 2 no jobs), $(0,2,0)$, $(0,0,2)$, $(1,1,0)$, $(1,0,1)$, and $(0,1,1)$. Note that the routing matrix P is not given, so that we cannot compute the visit ratios. Hence I leave them unspecified. Now, realize that for a fixed $\vec{n} = (n_0, n_1, n_2)$, $\Pi_i f_i(n_i) = f_0(n_0) f_1(n_1) f_2(n_2)$, so that if

$$\begin{aligned} \vec{n} &= (2,0,0) & f_0(2) f_1(0) f_2(0) &= \left(\frac{V_0}{\mu_0}\right)^2, \\ \vec{n} &= (0,0,2) & f_0(0) f_1(0) f_2(2) &= \left(\frac{V_2}{\mu_2}\right)^2, \\ \vec{n} &= (0,2,0) & f_0(0) f_1(2) f_2(0) &= \frac{1}{2}\left(\frac{V_1}{\mu_1}\right)^2. \end{aligned}$$

Note that in this last result we use that Station 1 has two servers. For the other combinations,

$$\begin{aligned} \vec{n} &= (1,1,0) & f_0(1) f_1(1) f_2(0) &= \frac{V_0}{\mu_0}\frac{V_1}{\mu_1}, \\ \vec{n} &= (1,0,1) & f_0(1) f_1(0) f_2(1) &= \frac{V_0}{\mu_0}\frac{V_2}{\mu_2}, \\ \vec{n} &= (0,1,1) & f_0(0) f_1(1) f_2(1) &= \frac{V_1}{\mu_1}\frac{V_2}{\mu_2}. \end{aligned}$$

Finally, add up all the above numbers to make $G(M,N)$:

Solution 4.2.1 It follows directly from the fact that $G(0,N)$ is a normalization constant. When there is just one station, i.e., station 0, all jobs are at that station. Hence, $\pi_0(N) = 1$. From Zijm.Eq.3.1–Zijm.Eq.3.3, it follows that $f_0(N) = (V_0/\mu_0)^N$, so that $G(0,N)$ must be $1/f_0(N)$. Finally, since $V_0 = 1$ always, we get $G(0,N) = \mu_0^{-N}$.

Solution 4.2.2 TBD.

Solution 4.2.3 This is related to the famous ball/boxes problem in Feller. Given n (identical) balls and m boxes, in how many configurations can the balls be distributed over the boxes? One nice way to get the answer is to use recursion. (I leave it to you to figure this out). The other is this: imagine that all boxes are put next to each other. Then we can imagine that two neighboring boxes are separated by a stick. Since there are m boxes, $m-1$ sticks suffice to separate all boxes. This observation allows us to reduce the balls/boxes problem to figuring out in how many ways all sticks and balls can be mixed. Since the balls and sticks are all identical, and since we have $n+m-1$ objects (n balls and $m-1$ sticks), these objects can be distributed in

$$\binom{n+m-1}{n} = \binom{n+m-1}{m-1}$$

ways. (BTW, this question relates directly to the number of edge points that occur as potential solutions in a linear programming problem.)

Now interpret the boxes as stations and the jobs as balls. Since we have $M+1$ stations, we get the result.

Solution 4.2.4 If the production rate at station i is infinitely large, the time a job stays at station i must be 0. Thus, the only 'activity' of station i would be to distribute all arriving jobs between its receiving stations, without any delay.

Thus, in both cases (i.e., setting the rate to infinity or defining $P'_{j,k} = P_{ji}P_{ik} + P_{j,k}$) it comes down to 'short-circuiting' node i.

Solution 4.2.5 From Zijm.Eq.3.3 for $c_i = 1$ we see that $f_i(n_i) = (V_i/\mu_i)^{n_i}$, since $\min\{c_i,k\} = \min\{1,k\} = 1$. Now using Zijm.Eqs 3.5 and 3.6,

$$\begin{aligned}
\mathbb{P}\{n_i \geq k\} &= \sum_{m=k}^{N} \pi(n_i = m) \\
&= \frac{1}{G(M,N)} \sum_{m=k}^{N} f_i(m) G(M \setminus \{i\}, N-m) \\
&= \frac{1}{G(M,N)} \sum_{m=k}^{N} \left(\frac{V_i}{\mu_i}\right)^m G(M \setminus \{i\}, N-m) \\
&= \frac{1}{G(M,N)} \left(\frac{V_i}{\mu_i}\right)^k \sum_{m=0}^{N-k} \left(\frac{V_i}{\mu_i}\right)^m G(M \setminus \{i\}, N-k-m) \\
&= \frac{f_i(k)}{G(M,N)} \sum_{m=0}^{N-k} \left(\frac{V_i}{\mu_i}\right)^m G(M \setminus \{i\}, N-k-m) \\
&= \frac{f_i(k)}{G(M,N)} G(M, N-k).
\end{aligned}$$

For $c > 1$, the expression for f_i is less simple, and the above derivation is not possible anymore.

Solution 4.2.6 The last time I solved this problem was in 2001. My answer below reflects my train of thought, including the mistakes.

My first guess is that $\pi_i(1) = \pi((0,0,\ldots,1,\ldots,0)) = 1/M$, where $(0,0,\ldots,1,\ldots,0)$ is the vector with a 1 at the i-th position and 0 elsewhere. It must be that $\pi(n) = 0$ for $n > 1$, because there is just one job in the system.

Let us check Zijm.Eq.3.5:

$$\pi(n_i = k) = f_i(k)\frac{G(M \setminus \{i\}, N-k)}{G(M,n)},$$

and Zijm.3.3:

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

We take $N = 1$ in Zijm.3.3. Since $\sum_i n_i = N$, and $N = 1$, $n_i \in \{0,1\}$. Thus,

$$f_i(0) = 1.$$

This is a bit tricky, since the product Π over k is empty. Such empty products are defined to be 1. Next, whatever c_i, $\min\{1, c_i\} = 1$. Thus,

$$f_i(1) = \frac{1}{1}\left(\frac{V_i}{\mu_i}\right)^1.$$

Now we need the visit ratios V_i. The visit ratios must be one since all jobs move from station n to $n+1$, in a circle. BTW, this is the same as Zijm.Question 3.1.2. Thus,

$$f_i(1) = \frac{1}{\mu_i}.$$

Now I suddenly see that my initial guess about $\pi_i(1) = \pi(n_i = 1)$ must be wrong. Rather, it has to be proportional to the service times! So, I'll revise my guess to $\pi_i(1) = \mu_i^{-1}/\sum_{j=0}^{M} \mu_i^{-1}$.

I also see now I can better use Zijm.3.6:

$$G(M,N) = \sum_{k=0}^{N} f_i(k) G(M \setminus \{i\}, N-k).$$

So, $G(M,1) = f_1(0)G(M \setminus \{1\},1) + f_1(1)G(M \setminus \{1\},0)$. Now, when there is no job in the network, there can be only configuration, hence $G(M,0) = 1$ for all M. Thus, $G(M,1) = G(M \setminus \{1\},1) + 1/\mu_1$, where I use the above results for $f_i(0)$ and $f_i(1)$. But this holds for any station i, so

$$G(M,1) = G(M \setminus \{1\},1) + 1/\mu_1 = G(M \setminus \{i\},1) + 1/\mu_i,$$

and so the only choice left for $G(M,N) = \sum_i \mu_i^{-1}$, just as I guessed after my repair.

I still need to fill in Zijm.3.5. From the above I learn that $f_i(0)G(M \setminus \{i\}, 1-0) = G(M \setminus \{i\},1)$ and $f_i(1)G(M \setminus \{i\},0) = \mu_i^{-1}$. Thus,

$$\pi_i(0) = f_i(0)\frac{G(M \setminus \{i\}, 1-0)}{G(M,1)} = \frac{G(M \setminus \{i\},1)}{G(M,N)} = \frac{\sum_{j \neq i} \mu_m^{-1}}{\sum_{j=1}^{M} \mu_j^{-1}}.$$

$$\pi_i(1) = f_i(1)\frac{G(M \setminus \{i\},0)}{G(M,1)} = \frac{\mu_i^{-1}}{G(M,N)} = \frac{\mu_i^{-1}}{\sum_{j=1}^{M} \mu_j^{-1}}.$$

I am happy to see that my second, less naive, guess comes out right. So why then must this be the intuitive answer? Lets consider a trivial network, with one load/unload node, and two single-server stations, one with a very fast server, and the other with a very slow server. Then the job must be at the slow server most of the time. The most natural guess is then that the time spent at a station, provided the visit ratios are one and there is one job, is proportional to its service time.

Solution 4.2.7 $G(2,2)$ means that we have 2 stations and 2 jobs. Lets start from Zijm.Eq.3.6 but now applied to node $M = 2$:

$$G(2,2) = \sum_{k=0}^{2} f_2(k) G(2 \setminus \{2\}, 2-k),$$

and use Zijm.Eq.3.13

$$G(0,2) = \mu_0^{-2}.$$

$$\begin{aligned} G(2,2) &= \sum_{k=0}^{2} f_2(k) G(2 \setminus \{2\}, 2-k) = \sum_{k=0}^{2} f_2(k) G(1, 2-k) \\ &= f_2(0)G(1,2) + f_2(1)G(1,1) + f_2(2)G(1,0) \\ &= G(1,2) + \mu_2^{-1} G(1,1) + \mu_2^{-2} G(1,0) \end{aligned}$$

where we use Zijm.Eq.3.3 to see that $f_2(k) = \mu_2^{-k}$ (realize again that we took the visit ratios to be one). Thus, we need $G(1,\cdot)$.

$$\begin{aligned} G(1,2) &= \sum_{k=0}^{2} f_1(k) G(1 \setminus \{1\}, 2-k) = \sum_{k=0}^{2} f_1(k) G(0, 2-k) \\ &= f_1(0)G(0,2) + f_1(1)G(0,1) + f_1(2)G(0,0), \\ &= f_1(0)\mu_0^{-2} + f_1(1)\mu_0^{-1} + f_1(2), \\ &= \mu_0^{-2} + \mu_1^{-1}\mu_0^{-1} + \mu_1^{-2}, \end{aligned}$$

where we use that $f_1(k) = \mu_1^{-k}$ and $G(M,0) = 1$ and $G(0,N) = \mu_0^{-N}$. Next,

$$\begin{aligned} G(1,1) &= f_1(0)G(0,1) + f_1(1)G(0,0) = \mu_0^{-1} + \mu_1^{-1}, \\ G(1,0) &= 1. \end{aligned}$$

Filling in the above, we get

$$G(2,2) = \mu_0^{-2} + \mu_1^{-1} + \mu_2^{-2} + (\mu_0\mu_1)^{-1} + (\mu_1\mu_2)^{-1} + (\mu_0\mu_2)^{-1}.$$

Solution 4.2.8 With the code I can check the results of the previous problem. I'll discuss the code in steps. As always, you can skip the code, but I would encourage you to study this example very carefully. Your computational skills will improve quite a bit.

We first need a few libraries.

```
from functools import lru_cache
import numpy as np
from numpy.linalg import eig
```

Since the convolution algorithm is recursive I use caching to store the results of intermediate computations. The storage is taken care of by the, so-called, decorator `lru_cache`. This programming concept is typically called *memoization*; there is a tutorial in R on memoization with Fibonacci sequences (see google) if you prefer to study this idea in R. Memoization is an important concept to learn as it can give an enormous computational speed up, often from exponential to linear complexity..

The data is this:

```
mu = np.array([2, 1, 3])
c = np.array([1, 1, 1]) # single server stations
P = np.matrix([
    [0, 1, 0],
    [0, 0, 1],
    [1, 0, 0]
]
)
```

I now compute the visit ratios V. This vector is the left eigenvector of the routing matrix P with eigenvalue 1, i.e., $V = VP$. Since P is a routing matrix it can be proven that only one eigenvalue is equal to 1, and that for all the other eigenvalues the real part is less than 1, that is, if v is such an eigenvalue, then $\Re(v) < 1$. (If you are interested to understand why, check the Perron-Frobenius theorem.) Thus, the visit ratio is the only eigenvector related to the eigenvalue 1, and, moreover, the eigenvalue 1 is the largest of all eigenvalues.

Compute the eigenvalues and eigenvectors:

```
v, w = eig(P.T)
```

Get the index of the eigenvalue with the largest real part, i.e., the eigenvalue 1:

```
x = v.real.argmax()
```

V is the eigenvector with eigenvalue 1; take the real part of the entries of V:

```
V = w[:,x].real
```

Normalize so that $V_0 = 1$.

```
>>> V = V / V[0]
>>> V
matrix([[ 1.],
        [ 1.],
        [ 1.]])
```

This is what we expected.

It follows from Zijm.Eq.3.3 that

$$f_i(n_i) = \begin{cases} \frac{V_i}{\mu_i} \frac{1}{\min\{n_i, c_i\}} f_i(n_i - 1), & \text{if } n_i > 0, \\ 1, & \text{if } n_i = 0. \end{cases}$$

```
@lru_cache(maxsize=None)
def f(i, n_i):
```

```
    if n_i == 0:
        return 1
    return V[i] / mu[i] / min(n_i, c[i]) * f(i, n_i - 1)
```

Note that the cache stores the results of intermediate computations.

Finally the convolution algorithm.

```
@lru_cache(maxsize=None)
def G(m, n):
    if n == 0:
        return 1
    if m == 0:
        return 1 / mu[0]**n
    return sum(f(m, k) * G(m - 1, n - k) for k in range(n + 1))
```

Observe how simple this implementation is compared to the specification in the text. I also find it much easier to read.

The result of the previous example must be

```
>>> G(2, 2)
matrix([[ 2.36111111]])
```

Testing code is crucial, just as crucial as testing formulas. The analytic result of the previous question becomes in code:

```
def test():
    res = 1 / mu[0]**2 + 1 / mu[1]**2 + 1 / mu[2]**2
    res += 1 / mu[0] * 1 / mu[1] + 1 / mu[0] * \
        1 / mu[2] + 1 / mu[1] * 1 / mu[2]
    print(res)

test()
```

```
2.36111111111
```

The outcomes of both procedures are the same.

When M and N are in the order of 1000, then I guess that the algorithm will not be that useful anymore. Even model the production situation as a closed queueing network will, typically, use its value. For instance, the convolution algorithm requires the processing rates of all the stations. Who on earth will assemble the processing rates of 1000 machines? Often, only the machines with the highest load are important the analyze, as these machines are the bottleneck. The rest can be, more or less, neglected, or modeled with very simple models.

Solution 4.2.9 Because then the form of $f_i(n_i) \neq (V_i/\mu_i)^{n_i}$, but much less simple.

We can try the recursion of Example 3.3.

```
mu = np.array([2, 1, 3])
V = np.array([1, 1, 1])


@lru_cache(maxsize=None)
def G(M, N):
    if N == 0:
        return 1
```

```
    if M == 0:
        return 1 / mu[M]**N
    return G(M - 1, N) + G(M, N - 1) * V[M] / mu[M]

print(G(2, 2))
```

```
2.36111111111
```

Solution 4.3.1 From $VP = V$ we conclude that $V_1 = 2V_0$. Take $n = 1$.

$$\begin{aligned}
\mathbb{E}[W_0(1)] &= \mathbb{E}[L_0(0)+1]\,\mathbb{E}[S_0] = \mathbb{E}[S_0] = 2,\\
\mathbb{E}[W_1(1)] &= \mathbb{E}[L_1(0)+1]\,\mathbb{E}[S_1] = \mathbb{E}[S_1] = 3,\\
\mathbb{E}[W(1)] &= \sum_{i=0}^{1} V_i\,\mathbb{E}[W_i(1)] = V_0 2 + V_1 3 = 1\cdot 2 + 2\cdot 3 = 8\\
\mathrm{TH}_0(1) &= \frac{1}{\mathbb{E}[W(1)]} = \frac{1}{8},\\
\mathbb{E}[L_0(1)] &= \mathrm{TH}_0(1)\mathbb{E}[W_0(1)] = \frac{2}{8} = \frac{1}{4}\\
\mathbb{E}[L_1(1)] &= \mathrm{TH}_1(1)\mathbb{E}[W_1(1)] = V_1\mathrm{TH}_0(1)\mathbb{E}[W_1(1)] = 2\cdot\frac{1}{8}\cdot 3 = \frac{3}{4}.
\end{aligned}$$

Now take $n = 2$.

$$\begin{aligned}
\mathbb{E}[W_0(2)] &= (\mathbb{E}[L_0(1)+1]\,\mathbb{E}[S_0] = (1/4+1)\mathbb{E}[S_0] = \frac{5}{4}\cdot 2 = \frac{5}{2},\\
\mathbb{E}[W_1(2)] &= (\mathbb{E}[L_1(1)+1]\,\mathbb{E}[S_1] = (3/4+1)\mathbb{E}[S_1] = \frac{7}{4}\cdot 3 = \frac{21}{4},\\
\mathbb{E}[W(2)] &= \sum_{i=0}^{1} V_i\,\mathbb{E}[W_i(2)] = \frac{5}{2} + 2\cdot\frac{21}{4} = 13\\
\mathrm{TH}_0(2) &= \frac{2}{\mathbb{E}[W(2)]} = \frac{2}{13},\\
\mathrm{TH}_1(2) &= V_1\mathrm{TH}_0(2) = \frac{4}{13},\\
\mathbb{E}[L_0(2)] &= \mathrm{TH}_0(2)\mathbb{E}[W_0(2)] = \frac{2}{13}\cdot\frac{5}{2} = \frac{5}{13}\\
\mathbb{E}[L_1(2)] &= \mathrm{TH}_1(2)\mathbb{E}[W_1(2)] = \frac{4}{13}\cdot\frac{21}{4} = \frac{21}{13}.
\end{aligned}$$

And so on.

Solution 4.3.2 Since there are quite a lot of jobs, and station 0 is the bottleneck, i.e., the station with the highest load, most of the time there will be a queue at station 0. In fact, most of the jobs will be in queue in front of station 0. Therefore the rate out of station 0 will be approximately equal to its service rate, i.e., μ_0. Thus, station 1 will receive jobs at a rate $\lambda_1 \approx \mu_0$, i.e., $\lambda_1 \approx 2$. Now I simply approximate the queueing process at station 1 as an $M/M/1$ queue with $\mu_1 = 2.5$. Hence, $\rho_1 = \lambda_1/\mu_1 = 2/2.5 = 4/5$, hence $L_1 = \rho_1/(1-\rho_1) = (4/5)/(1/5) = 4$.

Solution 4.3.3 I use the MVA algorithm to compute the expected number of jobs at each of the stations. Again, as in an earlier question in which I implemented the convolution algorithm, nearly as a one-liner, studying the code in detail is very rewarding. I include the numerical results at the end.

Start with computing the visit ratios.

```
from functools import lru_cache
import numpy as np
from numpy.linalg import eig

mu = np.array([2, 2.5, 3, 3, 3])
c = np.array([1, 1, 1, 1, 1])  # single server stations
P = np.matrix([
    [0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0],
    [0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0],
]
)

v, w = eig(P.T)
x = v.real.argmax()
V = w[:, x].real
V = V / V[0]
print(V)
```

```
[[ 1.]
 [ 1.]
 [ 1.]
 [ 1.]
 [ 1.]]
```

The visit ratios are according to expectation.

The following implements the MVA algorithm. Note how easy the code becomes with memoization; I only have to specify the recursions and the 'boundary conditions', i.e., what to do when $n = 0$. For the rest I don't have to think about in what exact sequence each of the functions needs to be called. The memoization takes care of all these problems..

```
@lru_cache(maxsize=None)
def EL(j, n):
    if n <= 0:
        return 0
    else:
        return TH(j, n) * EW(j, n)


@lru_cache(maxsize=None)
def EW(j, n):
    return (EL(j, n - 1) + 1) / mu[j]


@lru_cache(maxsize=None)
def TH(j, n):
    if j == 0:
```

```
        return n / sum(V[j] * EW(j, n) for j in range(len(mu)))
    else:
        return V[j] * TH(0, n)
```

This is all!

Now I print the expected jobs at the stations.

```
print("  n    EL0    EL1    EL2   EL3    EL4    TH0")
for n in range(1,30):
    res = "{:3d}".format(n)
    for j in range(len(mu)):
        res += "{:6.2f}".format(float(EL(j, n)))
    res += "{:6.2f}".format(float(TH(0,n)))
    print(res)
```

```
  n    EL0    EL1    EL2   EL3    EL4    TH0
  1   0.26   0.21   0.18   0.18   0.18   0.53
  2   0.55   0.42   0.34   0.34   0.34   0.87
  3   0.87   0.64   0.50   0.50   0.50   1.12
  4   1.21   0.85   0.65   0.65   0.65   1.30
  5   1.58   1.06   0.79   0.79   0.79   1.43
  6   1.99   1.27   0.92   0.92   0.92   1.54
  7   2.42   1.47   1.04   1.04   1.04   1.62
  8   2.89   1.67   1.15   1.15   1.15   1.69
  9   3.39   1.86   1.25   1.25   1.25   1.74
 10   3.93   2.05   1.34   1.34   1.34   1.79
 11   4.50   2.23   1.42   1.42   1.42   1.83
 12   5.10   2.40   1.50   1.50   1.50   1.86
 13   5.74   2.56   1.57   1.57   1.57   1.88
 14   6.41   2.70   1.63   1.63   1.63   1.90
 15   7.11   2.84   1.68   1.68   1.68   1.92
 16   7.84   2.97   1.73   1.73   1.73   1.93
 17   8.60   3.09   1.77   1.77   1.77   1.95
 18   9.39   3.20   1.80   1.80   1.80   1.96
 19  10.20   3.30   1.84   1.84   1.84   1.96
 20  11.03   3.39   1.86   1.86   1.86   1.97
 21  11.88   3.47   1.88   1.88   1.88   1.98
 22  12.75   3.54   1.90   1.90   1.90   1.98
 23  13.64   3.60   1.92   1.92   1.92   1.98
 24  14.54   3.66   1.93   1.93   1.93   1.99
 25  15.46   3.70   1.95   1.95   1.95   1.99
 26  16.39   3.75   1.96   1.96   1.96   1.99
 27  17.33   3.78   1.96   1.96   1.96   1.99
 28  18.27   3.82   1.97   1.97   1.97   1.99
 29  19.23   3.84   1.98   1.98   1.98   2.00
```

This is interesting. Using the insights of the previous question, approximation stations 3, 4 and 5 as $M/M/1$ queues, we have according to this model that $\mathbb{E}[L_3] = \rho_3/(1-\rho_3)$. Taking $\lambda_3 = 2$ and $\mu_3 = 3$, we see that $\rho_3 = 2/3$ so that $\mathbb{E}[L_3] = 2$. This is very close to 1.98.

For the last line, with $n = 29$, the number of jobs at station 0 must therefore be approximately $29-4-2-2-2 = 19$. The algorithm supports our intuition!

Solution 4.3.4 See the answer of the previous question.

Solution 4.4.1 Take $n = 1$.

$$\begin{aligned}
\mathbb{E}[W_0(1)] &= \sum_{k=c_0}^{n-1} \frac{k-c_0+1}{c_0\mu_0} p_0(k|1-1) + \frac{1}{\mu_0} \\
&= \sum_{k=1}^{0} \frac{k-c_0+1}{c_0\mu_0} p_0(k|1-1) + \frac{1}{\mu_0} \\
&= \frac{1}{\mu_0} \\
&= 2, \\
\mathbb{E}[W_1(1)] &= \sum_{k=c_1}^{n-1} \frac{k-c_1+1}{c_1\mu_1} p_1(k|1-1) + \frac{1}{\mu_1} \\
&= \sum_{k=2}^{0} \frac{k-c_1+1}{c_1\mu_1} p_1(k|1-1) + \frac{1}{\mu_1} \\
&= \frac{1}{\mu_1} \\
&= 3, \\
\mathbb{E}[W(1)] &= \sum_{i=0}^{1} V_i \mathbb{E}[W]_i(1) = V_0 2 + V_1 3 = 1\cdot 2 + 2\cdot 3 = 8 \\
\mathrm{TH}_0(1) &= \frac{1}{\mathbb{E}[W(1)]} = 1/8, \\
\mathrm{TH}_1(1) &= V_1 \mathrm{TH}_0(1) = 2/8 = 1/4, \\
p_0(1|1) &= \mathrm{TH}_0(1) p_0(0|0) \mathbb{E}[S_0] \frac{1}{\min\{c_0,1\}} = \frac{1}{8}\cdot 1\cdot 2\cdot \frac{1}{1} = 1/4, \\
p_1(1|1) &= \mathrm{TH}_1(1) p_1(0|0) \mathbb{E}[S_1] \frac{1}{\min\{c_1,1\}} = \frac{1}{4}\cdot 1\cdot 3\cdot \frac{1}{1} = 3/4, \\
p_0(0|1) &= 1 - p_0(1|1) = 3/4, \\
p_1(0|1) &= 1 - p_1(1|1) = 1/4,
\end{aligned}$$

Now for $n = 2$.

$$\begin{aligned}
\mathbb{E}[W_0(2)] &= \sum_{k=c_0}^{n-1} \frac{k-c_0+1}{c_0\mu_0} p_0(k|2-1) + \frac{1}{\mu_0} \\
&= \sum_{k=1}^{1} \frac{k-c_0+1}{c_0\mu_0} p_0(k|1) + \frac{1}{\mu_0} \\
&= \sum_{k=1}^{1} \frac{k-c_0+1}{c_0} \mathbb{E}[S_0] p_0(k|1) + \frac{1}{\mu_0} \\
&= \frac{1-1+1}{1} 2p_0(1|1) + 2 \\
&= 2 \cdot 1/4 + 2 = 5/2 \\
\mathbb{E}[W_1(2)] &= \sum_{k=2}^{2-1} \frac{k-c_1+1}{c_1\mu_1} p_1(k|1-1) + \frac{1}{\mu_1} \\
&= \frac{1}{\mu_1} = 3, \\
\mathbb{E}[W(2)] &= \sum_{i=0}^{1} V_i \mathbb{E}[W_i(2)] = 1 \cdot 5/2 + 2 \cdot 3 = 5/2 + 6 = 17/2 \\
\mathrm{TH}_0(2) &= \frac{2}{\mathbb{E}[W(2)]} = 2 \cdot 2/17 = 4/17, \\
\mathrm{TH}_1(2) &= V_1 \mathrm{TH}_0(2) = 2 \cdot 4/17 = 8/17, \\
p_0(1|2) &= \mathrm{TH}_0(2) p_0(0|1) \mathbb{E}[S_0] \frac{1}{\min\{c_0, 1)} \\
&= 4/17 \cdot 3/4 \cdot 2 \cdot 1 \\
p_0(2|2) &= \mathrm{TH}_0(2) p_0(1|1) \mathbb{E}[S_0] \frac{1}{\min\{c_0, 1)} \\
&= 2/17 \cdot 1/4 \cdot 2 \cdot 1 \\
p_0(0|2) &= 1 - p_0(1|2) - p_0(2|2) \\
p_1(1|2) &= \mathrm{TH}_1(2) p_1(0|1) \mathbb{E}[S_1] \frac{1}{\min\{c_1, 1)} \\
&= 8/17 \cdot 1/4 \cdot 3 \cdot 1/1 \\
p_1(2|2) &= \mathrm{TH}_1(2) p_1(1|1) \mathbb{E}[S_1] \frac{1}{\min\{c_1, 2)} \\
&= 8/17 \cdot 3/4 \cdot 3 \cdot 1/2 \\
p_1(0|2) &= 1 - p_1(1|2) - p_1(2|2).
\end{aligned}$$

I am done (and fed up). Its time to for the computer to take over ...

Solution 4.4.2 There is no end to the number of variations we can make. An interesting case is to study the effect of increasing the capacity of the second slowest station. When the total number of jobs in the network is large, nearly all of them will wait in the queue in front of the bottleneck, here station 0. If, however, the number is not large, it may be interesting to increase the rate of the second slowest station (if that is easy or cheap compared to increasing the production rate of the bottleneck). Then the queue at the second slowest station will become smaller (at least, that is what I expect), hence the queue in front of the bottleneck must increase

(I don't expect that the jobs from the second slowest will move to the faster stations.) But then the fraction of idle time at the bottleneck must go down, hence the throughput must increase. Note that, if the throughput increases, the arrival rate at the other stations must increase, hence, the queue lengths at these stations must also increase. Thus effects runs counter the other effect (increasing the queue at the bottleneck.)

So lets try.

First the old results again:

```
from functools import lru_cache
import numpy as np
from numpy.linalg import eig


c = np.array([1, 1, 1, 1, 1])  # single server stations
P = np.matrix([
    [0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0],
    [0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0],
]
)

v, w = eig(P.T)
x = v.real.argmax()
V = w[:, x].real
V = V / V[0]

mu = np.array([2, 2.5, 3, 3, 3])

@lru_cache(maxsize=None)
def EL(j, n):
    if n <= 0:
        return 0
    else:
        return TH(j, n) * EW(j, n)

@lru_cache(maxsize=None)
def EW(j, n):
    return (EL(j, n - 1) + 1) / mu[j]


@lru_cache(maxsize=None)
def TH(j, n):
    if j == 0:
        return n / sum(V[j] * EW(j, n) for j in range(len(mu)))
    else:
        return V[j] * TH(0, n)

print(" n  EL0  EL1  EL2   EL3   EL4 TH0")
for n in range(8, 11):
    res = "{:3d}".format(n)
    for j in range(len(mu)):
        res += "{:6.2f}".format(float(EL(j, n)))
    res += "{:6.2f}".format(float(TH(0,n)))
    print(res)
```

```
 n   EL0   EL1   EL2    EL3    EL4  TH0
  8   2.89   1.67   1.15   1.15   1.15   1.69
  9   3.39   1.86   1.25   1.25   1.25   1.74
 10   3.93   2.05   1.34   1.34   1.34   1.79
```

Now we need to clear the cache, and redo the computations The visit ratios do not change, only the processing rates.

```
EL.cache_clear()
EW.cache_clear()
TH.cache_clear()

mu = np.array([2, 3, 3, 3, 3])

print(" n   EL0   EL1   EL2    EL3    EL4  TH0")
for n in range(8, 11):
    res = "{:3d}".format(n)
    for j in range(len(mu)):
        res += "{:6.2f}".format(float(EL(j, n)))
    res += "{:6.2f}".format(float(TH(0,n)))
    print(res)
```

```
 n   EL0   EL1   EL2    EL3    EL4  TH0
  8   3.14   1.21   1.21   1.21   1.21   1.74
  9   3.71   1.32   1.32   1.32   1.32   1.79
 10   4.32   1.42   1.42   1.42   1.42   1.83
```

All is according to intuition. Indeed, the throughput increases slightly. Thus, it may be interesting to increase the production rate of non-bottleneck machines, provided its (much) easier/cheaper than improving the bottleneck.

Solution 4.4.3 I first implement the algorithm. Then I use the case, i.e., the data of Example 3.4, and compare, as a test, the results to the results of the previous question.

```
from functools import lru_cache
import numpy as np
from numpy.linalg import eig

mu = np.array([2, 2.5, 3, 3, 3])
c = np.array([1, 1, 1, 1, 1])  # single server stations
P = np.matrix([
    [0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0],
    [0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0],
]
)

v, w = eig(P.T)
x = v.real.argmax()
V = w[:, x].real
V = V / V[0]
```

```
@lru_cache(maxsize=None)
def p(j, k, n):
    if n == 0:
        return 1
    if k > 0:
        return TH(j, n) * p(j, k - 1, n - 1) / min(c[j], k) / mu[j]
    else:
        return 1 - sum(p(j, k, n) for k in range(1, n + 1))


@lru_cache(maxsize=None)
def EW(j, n):
    res = sum((k - c[j] + 1) / c[j] / mu[j] * p(j, k, n - 1)
              for k in range(c[j], n))
    res += 1 / mu[j]
    return res


@lru_cache(maxsize=None)
def TH(j, n):
    if j == 0:
        return n / sum(V[j] * EW(j, n) for j in range(len(c)))
    else:
        return V[j] * TH(0, n)

print(TH(0, 10))
```

```
[[ 1.78901276]]
```

This is the same as what we see earlier. I guess all is ok.

Now add a machine to station 0. Don't forget to clear the cache!

```
>>> p.cache_clear()
>>> EW.cache_clear()
>>> TH.cache_clear()
>>>
>>> c = np.array([2, 1, 1, 1, 1])
>>>
>>> n = 10
>>> TH(0, n)
matrix([[ 2.08301549]])
>>> n = 30
>>> TH(0, n)
matrix([[ 2.46519749]])
```

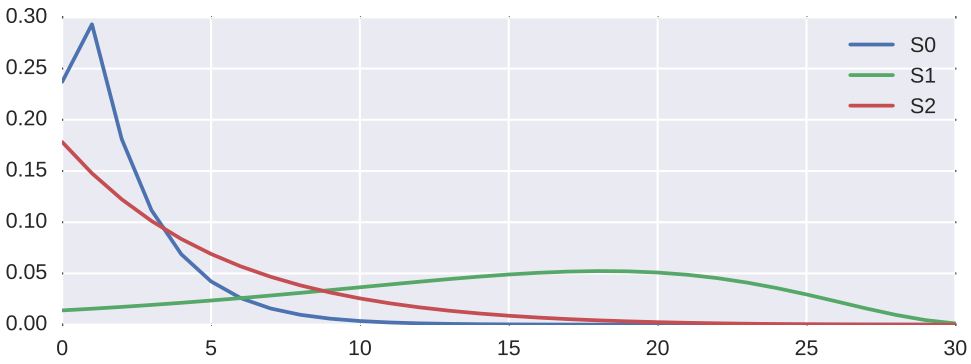
As expected, station 1 with processing rate $\mu_1 = 2.5$ now determines the throughput of the network.

```
def ELQ(j, n):
    return sum((k - c[j]) * p(j, k, n) for k in range(c[j], n + 1))


def EL(j, n):
    return sum(k * p(j, k, n) for k in range(n + 1))

print(EL(0, n))
print(TH(0, n) * EW(0, n)) # Little's law as a check.
```

```
[[ 1.9724301]]
[[ 1.9724301]]
```

Finally, plot the probability mass functions for stations 0, 1, and 2.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_context("paper")

plt.figure(figsize=(6, 2))
plt.plot([float(p(0, k, n)) for k in range(n + 1)], label="S0")
plt.plot([float(p(1, k, n)) for k in range(n + 1)], label="S1")
plt.plot([float(p(2, k, n)) for k in range(n + 1)], label="S2")
plt.legend()
plt.show()
filename = "figures/mda_mass.pdf"
plt.savefig(filename)
plt.close()
```



7 Notation

a_n = Number of arrivals in the nth period

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

A_k = Arrival time of kth job

c_n = Service/production capacity in the nth period

d_n = Number of departures in the nth period

c = Number of servers

C_a^2 = Squared coefficient of variation of the interarrival times

C_s^2 = Squared coefficient of variation of the service times

$D(t)$ = Number of departures in $[0,t]$

$D_Q(t)$ = Number of customers/jobs that departed from the queue in $[0,t]$

D_n = Departure time of nth job

$D_{Q,n}$ = Departure time of the queue of nth job

F = Distribution of the service time of a job

$L(t)$ = Number of customers/jobs in the system at time t

$Q(t)$ = Number of customers/jobs in queue at time t

$L_S(t)$ = Number of customers/jobs in service at time t

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

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

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

$N(t)$ = Number of events in $[0,t]$

$N(s,t)$ = Number of events in $(s,t]$

$p(n)$ = Long-run time average that the system contains n jobs

Q_n = Queue length as seen by the nth job, or at the *end* of the nth period

S_k = Service time required by the kth job

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

S = generic service time of a job

t = time

W_n = time in the system of nth job

$W_{Q,n}$ = time in the queue of nth job

$\mathbb{E}[W]$ = sample average of the sojourn time

$\mathbb{E}[W]_Q$ = sample average of the time in queue

X_k = inter-arrival time between job $k-1$ and job k

X = generic interarrival time between two consecutive jobs

γ = departure rate

λ = arrival rate

μ = service rate

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

ρ = Load on the system

8 Formula Sheet

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

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

$$C_e^2 = C_0^2 + 2A(1-A)\frac{m_r}{\mathbb{E}[S_0]}$$

$$\sigma_e^2 = \sigma_0^2 + \frac{\sigma_s^2}{N_s} + \frac{N_s - 1}{N_s^2}(\mathbb{E}[S_s])^2$$

$$\sigma_e^2 = \sigma_0^2 + \sigma_r^2 f_r + f_r(1-f_r)(\mathbb{E}[S_r])^2$$

$$C_{sB}^2 = \frac{\mathbb{V}[S]_s + N\mathbb{V}[S]_0}{(\mathbb{E}[S_s] + N\mathbb{E}[S_0])^2}$$

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

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

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

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

$$C_{ij}^2 = P_{ij}C_{di}^2 + 1 - P_{ij}$$

$$Q_{ij} = \frac{\lambda_{ij}}{\lambda_j}$$

$$C_{aj}^2 = w_j \sum_{i=0}^{M} Q_{ij}C_{ij}^2 + 1 - w_j$$

$$w_j = (1 + 4(1-\rho_i)^2(v_j - 1))^{-1}$$

$$v_j = \left(\sum_{i=0}^{M} Q_{ij}^2\right)^{-1}, j = 1, \ldots, M,$$

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

$$G(m,n) = \sum_{k=0}^{n} f_m(k)G(m-1, n-k)$$

$$G(m,0) = 1, G(0,n) = \mu_0^{-n},$$

$$\mathbb{E}[W]_j(n) = \mathbb{E}\left[L_j(n-1) + 1\right]\mathbb{E}[S]_j$$

$$\mathrm{TH}_0(n) = \frac{n}{\sum_{j=0}^{M} V_j \mathbb{E}\left[W_j(n)\right]}$$

$$\mathrm{TH}_j(n) = V_j \mathrm{TH}_0(n)$$

$$\mathbb{E}\left[L_j(n)\right] = \mathrm{TH}_j(n)\mathbb{E}\left[W_j(n)\right]$$

$$p_j(0|0) = 1$$

$$\mathbb{E}\left[W_j(n)\right] = \sum_{k=c_j}^{n-1} \frac{k - c_j + 1}{c_j\mu_j} p_j(k|n-1) + \frac{1}{\mu_j}$$

$$\mu_j \min\{c_j, k\} p_j(k|n) = \mathrm{TH}_j(n) p_j(k-1|n-1)$$

$$p_j(0|n) = 1 - \sum_{k=1}^{n} p_j(k|n).$$

Bibliography

F. Baccelli and W.A. Massey. A sample path analysis of the $M/M/1$ queue. *Journal of Applied Probability*, 26(2):418–422, 1988.

G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley & Sons, 2006.

D.R. Cox, editor. *Renewal Theory*. John Wiley & Sons Inc, New York, 1962.

M. El-Taha and S. Stidham Jr. *Sample-Path Analysis of Queueing Systems*. Kluwer Academic Publishers, 1998.

R.W. Hall. *Queueing Methods for Services and Manufacturing*. Prentice Hall, 1991.