Computer Systems Performance Evaluation

Mark Crovella

2007

Contents

Acknowledgements

This book benefitted greatly from the ideas in [LeBoudec], [Ross], and [Allen].

“From the moment I picked up your book until I laid it down, I was convulsed with laughter. Someday I intend reading it.”
—Groucho Marx

Part I

Background

Chapter 1

Background

1.1 What is Performance Evaluation?

Performance evaluation is concerned with making quantitative statements about the behavior of computer systems. In practical terms this can mean making predictions about response time, throughput, or other metrics based on knowledge of CPU, disk, or other resource demands.

To this end we'll use three types of tools: analytic (mathematical) models (mainly applied probability), simulation, and measurement/experimentation.

The goal of this course then is to provide two related sets of skills:

- *Practical* tools for making predictions about the performance of real or imagined computer systems; and
- *Insight* into how computer systems behave as we vary their workloads and designs.

The first goal emphasizes sound formulations for calculations and measurement (more often) and analysis (sometimes). The second goal emphasizes drawing conclusions from calculations (sometimes) and analysis (more often).

An example of the first goal would be: capacity planning for the upgrade of a banking information system. Examples of the second goal would be answering questions like: "What causes delays in computer systems?;" "Which is better, one fast server or two slower ones?;" and "What scheduling policy is most fair to all customers?"

Throughout we will used *models* to aid our discussion. We use the term model to apply to simplified versions of a system under consideration. A model could be a mathematical expression, or a set of equations that can be solved, or a simulation. Models necessarily leave out some details of the real system. As such, "all models are wrong, but some are useful."

1.2 Probability and its Uses

What is probability? First, make sure to distinguish two common uses of the word: 1) subjective confidence in an event's occurrence; 2) frequency of occurrence of an event. We are concerned with the latter.

> Suppose an experiment under consideration can be repeated any number of times, so that, in principle at least, we can produce a whole series of "independent trials under identical conditions" in each of which, depending on chance, a particular event A of interest either occurs or does not occur. Let n be the total number of experiments in the whole series of trials, and let $n(A)$ be the number of experiement in which A occurs. Then the ratio $n(A)/n$ is called the relative frequency of the event A. It turns out that the relative frequencies observed in different series of trials are virtually the same for large n, clustering about some constant $P[A]$, which is called the probability of the event A.
>
> From [Roz69].

This description captures the notion of probability intuitively. Note however that it is only an intuitive definition: we must use our judgement to verify or decide what is meant by "independent trials under identical conditions." [1]

1.2.1 The Use of Models

There two roles that data models play in computer systems research, and their purposes can be confused. To get there, let's ask what a model is, actually.

We will refer to a data model as a succinct description of a generative process that gives rise to an output (ie, a dataset) of interest. We'll just call them "models" in what follows.

We can come up with models two ways, starting from a real computer system or from measured data. These are "white box" and "black box," or "constructive" and "descriptive" approaches. In the first approach, one starts with a real system and simplifies it somehow - for example, one starts with a real network and constructs an *ns* simulation. This then becomes the model. This is often done, eg, in performance evaluation.

In the second approach, one starts with real data and "fits" a model to it, in the way one "fits" an story to a set of facts. The idea is that the model "could have" generated the observed data.

[1] Aside: People tend in general to be good at estimating probabilities of certain kinds of events. It seems that our cognitive capabilities have been shaped by evolution to provide us with excellent skills at assessing potential dangers and opportunities. Nonetheless, for the same reasons, these probability estimation skills may be fooled in striking ways (witness the popularity of various lotteries, and the fact that people fear plane travel more than car travel). For an interesting discussion, see Chapter 5 of "How the Mind Works," by Steven Pinker.

Ususally the model is a random one, and so we can speak quantitatively about whether the model "could have" generated the observed data. This approach is more often used in "measurement" papers.

Since it is clear in both cases that model outputs are not exact matches for reality (ie, our datasets) we can speak of "better" or "worse" models in terms of how closely model outputs agree with measured data.

Both approaches have drawbacks.

Drawbacks to descriptive approach: such approaches do not explain "why", "what if system changes". They can be difficult to interpret. Such models generally do not use all the available information (we may know something about how data was generated that is not reflected in model). As a result we have to choose models carefully, to encourage maximum interpretability.

Drawbacks to constructive approach: generalization is difficult (too many parameters), output may not match data in important respects.

These two approaches an be used together: working toward the middle.

That is: System $\rightarrow$ model $\leftarrow$ data. In this case we gain confidence when the two forms of analysis suggest the same model.

Turning back to the roles models play, the analogy of fitting a story to a set of facts becomes important. The problem becomes clear when one asks *why* a model is desired. Some motivations are:

- the description will be used to generate more data, perhaps by others (one can use a model as a source of data for testing or evaluating systems).

- the description has parameters that are interpretable (Interpretation: relating effects to causes)

- the description can be used to interpret real data (fitting a model to data results in parameter values that may be interpretable).

Example parameters that are interpretable: mean, variance, λ of an exponential RV, α (tail shape) of a power-law RV. These are interpretable in the sense that as these parameters change, we understand how something in the real system changes (like performance).

Interpreting data: here we try to understand what is happening in the underlying system by using the model to constrain the possibilities.

For more details see [SF03].

1.2.2 The Use of Probability Models

Probabilistic models are abstractions. We use probabilistic models because, in general, there are things we don't want to model, or don't know how to model.

Quote from Matheron:

> In a serious work ... an expression such as "this phenomenon is due to chance" constitutes simply, an elliptic form of speech. ... It really mean "everything occurs as if this phenomenon were due to chance," or, to be more precise: "To describe, or interpret or formalize this phenomenon, only probabilistic models have so far given good results."

It is important therefore to very sharply distinguish between the properties of a model and the properties of the data. Data properties are observable quantities by definition. The model may well contain assumptions that are not operationally testable or directly observable, and therefore cannot actually be said to apply to the data. For example, mean, variance, stationarity, independence, etc, are all properties of models, not data. Some of these concepts can be given operational interpretations, such as the mean, which we can identify with the long-run arithmetic average. However we must be careful not to assume that the data is "truly" stationary, or "truly" independent, etc. These are properties that only models can have.

Making a clear distinction between properties of models and data can help avoid much confusion, and for example endless debates about whether data "is stationary." Rather, a meaningful statement would be "a stationary model is useful for describing this data" or "this data is consistent with a stationary model for the problem at hand."

We use probabilistic models because they help us answer questions. There is no real use, or even meaning, for a probabilistic model apart from a set of questions that it can help to answer. (Remember that most of the phenomena we are concerned with are not really random; they are just incompletely specified.) Thus another class of debates which we seek to avoid surrounds whether a model is the "correct" one for a set of data. We cannot answer this question as posed. Rather we should ask: "Does this model give useful answers to the questions for which it is employed?" Keeping this distinction in mind will also help us avoid another set of endless debates.

For more details see [Mat89].

1.3 Time and Event Views

> There's a time and the time is now and it's right for me,
> It's right for me, and the time is now.
>
> – *Time and a Word,* Yes

Much of what we will study in this course concerns the distinction between *time* and *event* views of a system. The central issue is this: the behavior of computer systems is generally *defined* in terms of events. At the most fundamental level, computers are finite-state machines and their behavior is defined in terms of transitions between states. Even at higher levels (programming languages, or applications) a computer system's behavior is conceived and defined in terms of events (execution of instructions, execution of lines of a program, etc). And at even higher levels we generally specify system behavior in terms of events (what happens when a customer arrives at a queue, how a system responds to a request for service, etc).

But what people experience is the *time* view of the system. The time view is the state of the system at a random time. For most systems, we can treat the "random time" view of the system as representative of either what one one user sees over a long duration, or what a randomly chosen user sees at any instant. That is, the "random time" view is essentially the *performance* of the system.

Aside: The formal notion that links "a random time" with "what a random customer sees" is called *ergodicity.* We'll define ergodicity later, but it captures the idea that observing many separately operating "copies" of a system at any instant can yield the same answers as observing a single system over a period of time. We will usually assume that systems are ergodic, and therefore we will be concerned with the time view as our main way of representing system performance.

This distinction is often stated as being one of *event averages* versus *time averages.* This is a good way to think about it, but not a complete one. We are not just concerned with averages, but in fact the entire distribution of behaviors.

As explained above, we introduce probability to avoid specifying unimportant details. This turns our event view of the system ("from state A, if E happens, then go to state B") into a *conditional probability:* "from state A, go to state B with probability p." To make the connection clearer, we would usually state this as :

$$P[\text{ current state is B} \mid \text{last state is A}] = p.$$

In contrast, the time view is essentially an unconditioned view. It is represented by a statement like:

$$P[\text{ current state is B}] = q.$$

This tells us "the system spends q fraction of its time in state B" which is a very different and usually more useful statement than the conditional one.

This translation between the specification of a system in terms of events and the description of a system in terms of time is at the heart of many aspects of performance evaluation. We will use a number of tools to address this challenge, but the two main tools we will use are *Palm Calculus* and *Markov Chains*. Both are sets of techniques that allow us to start with conditional probabilities (the event view) and derive unconditional probabilities (the time view).

> “I just the other day got, an internet was sent by my staff at 10 o’clock in the morning on Friday and I just got it yesterday. Why?
>
> Because it got tangled up with all these things going on the internet commercially...
>
> They want to deliver vast amounts of information over the internet. And again, the internet is not something you just dump something on. It’s not a truck.
>
> It’s a series of tubes.
>
> And if you don’t understand those tubes can be filled and if they are filled, when you put your message in, it gets in line and its going to be delayed by anyone that puts into that tube enormous amounts of material, enormous amounts of material.”
>
> — U.S. Senator Ted Stevens

Chapter 2

Probability Refresher

2.1 Probability, Formally

Despite the difficulty of working with probability at the intuitive level, we can nonetheless work with a formal definition of probability that allows us to reason precisely about probabilities.[1] This formal definition follows.

Definition. For a sample space Ω a *probability measure P* is a function defined on all the subsets of Ω (the events) such that:

1. $P[\Omega] = 1$
2. For any event $A \subset \Omega$, $P[A] \geq 0$.
3. For any events $A, B \subset \Omega$ where $A \cap B = \emptyset$, $P[A \cup B] = P[A] + P[B]$.

Some things we can immediately conclude are as follows. You should make sure that you can use the definition to prove that these are true:

Example 1. $P[A] = 1 - P[\bar{A}]$.

Example 2. For any two sets A and B, $P[A \cup B] = P[A] + P[B] - P[A \cap B]$.

[1] It can be shown that the framework built up using this definition can capture an intuitive sense of probability; this fact is called the "law of large numbers".

2.2 Conditional Probability

Conditional probability refers to how probability changes if we restrict our attention to a subset of events. Here's an example:

> **Example 1.** Let the sample space Ω be the set of all days on which CS 470 meets. Assume that the probability that I give a test on any day is 0.3, and that the probability that a day is cloudy is 0.5.

Now, I might never give a test on a cloudy day. Or I might be more likely to give a test if the day is cloudy. Or the fact that it is cloudy might have no effect on whether I give a test.

We can give precise expressions for these possibilities using the notion of conditional probability. Let the event of a day with a test be T. Let the event of a cloudy day be C. Then $P[T|C]$ is "the conditional probability of event T given C".

How should we define $P[T|C]$? Intuitively, this should be a measure of the number of times a T event occurs given that a C event occurs ... that is, the proportion of cloudy days on which I give a test.

So, more generally, $P[A|B]$ is the probability of an event A given that only outcomes which are in event B are considered. That is, $P[A|B]$ is a probability measure in which the universe Ω has been restricted to event B.

This is a probability measure, so $P[B|B] = 1$. Now if B is a proper subset of Ω, then $P[B] < 1$. So we have rescaled our probability measure so that $P[B|B] = 1$. This leads to our definition:

> **Definition.** The conditional probability of an event A given that event B (having positive probability) is known to occur, is
>
> $$P[A|B] = \frac{P[A\&B]}{P[B]} \text{ where } P[B] > 0$$

Continuing our example, let's say that you notice that out of 24 class days, I give a test on 6 cloudy days and 3 clear days. Furthermore, 18 class days are cloudy. Let's assume that the fraction of cloudy days is equal to $P[C]$ and the fraction of clear days is equal to $P[\bar{C}]$.

What is $P[T|C]$? Answer: $P[C] = 18/24 = .75$, and $P[T\&C] = 6/24 = .25$; so $P[T|C] = .33$.

What is $P[T|\bar{C}]$? Answer: 3/24 / 6/24 = 3/6 = .5

What is the probability of a test on a random day? Answer: $P[T] = 9/24 = 3/8 = .375$

One way to think about conditional probability is that, by restricting attention to a subset of all possible events, you are really adding more information to the setting. That is, a conditional probability reflects an "updated" estimate of probability given some new information. In our example, knowing that today is cloudy "decreases" the probability of a test from .375 to .33. (Actually, it decreases your *estimate* of the probability of a test.) Likewise knowing that today is sunny "increases" the probability of a test from .375 to .5.

2.2.1 Uses of Conditional Probability

Often the conditional event is easier to calculate than the compound event. In fact, this is one of the main uses of conditional probability: it is easier to determine a bunch of conditional probabilities than it is to dctermine a compound probability.

Example 2: a bucket contains two red and three white balls. Calculate the probability that you would draw, one at a time and without replacement, two white balls.

Let A be the event of drawing a white ball on first draw, and B be the event of drawing a white ball on second draw. Then:

$$\begin{aligned}
P[B|A] &= P[A\&B]/P[A] \\
\text{So: } P[B\&A] &= P[A]P[B|A] \\
\text{Now: } P[A] &= 3/5 \\
\text{and: } P[B|A] &= 1/2 \\
\text{So: } P[B\&A] &= 3/10
\end{aligned}$$

Consider how you would have solved this problem in a more direct manner (*.e.g.,* by enumerating all 120 possible orderings of the 5 balls) and note how much more complicated that approach is.

2.3 Independence

Returning to Example 1: notice how the probability of a test changed when we learned something new (the weather). Of course, this might not always be the case.

Again, consider $P[T|C]$. Remember 18/24 days are cloudy. Let's say I give tests on 6 cloudy days and 2 sunny days. Then:

$$\begin{aligned} P[T|C] &= P[T\&C]/P[C] = 6/18 = .333 \\ \text{and: } P[T] &= 8/24 = .333 \end{aligned}$$

So $P[T|C] = P[T]$. Then for this case, I have not "told you anything" about test likelihood by telling you today is cloudy. More precisely, the type of weather doesn't affect the probability of a test. This is called *independence.* Two events A and B are independent if $P[A|B] = P[A]$.

Note that since $P[A|B] = P[A\&B]/P[B]$ then $P[A] \cdot P[B] = P[A\&B]$.

So, we conclude that if two events are independent, the probability of both occurring is the product of the individual probabilities. These are equivalent definitions of independence.

2.4 Random Variables

So far we've been considering events. In reality we are usually more interested in numeric values associated with events. When a random event has a numeric value we refer to it as a random variable. Technically we say that a random variable is a function that assigns a real number to each possible outcome in a sample space. Note that a single experiment (throwing a dart at a sheet of paper) may have many random variables associated with it (different measurements of the outcome).

Notation: Random variables use CAPITALS. Values use lowercase.

Now, we'd like to collect information about what values of the random variable are more probable than others.

Definition. The cumulative distribution function (CDF) F for a random variable X is equal to the probability measure for the event that consists of all possible outcomes with a value of the random variable X less than or equal to x, that is, $F(x) = P[X \leq x]$.

Some facts: the domain of the CDF is $-\infty$ to $+\infty$. $F(-\infty) = 0$. $F(\infty) = 1$. F is nondecreasing.

Question: What is the range of the CDF?

Exercise: Consider the roll of a die. The obvious random variable here is the number of points showing. Plot the CDF.

Exercise: Consider throwing a dart at a dartboard, with no "bullseye" so the dart hits anywhere on the dartboard with equal likelihood. Let the random variable be the distance the dart hits from the left edge of the paper. Plot the CDF.

Random variables can be discrete (taking on a finite or countably infinite set of values) or continuous (taking on an uncountably infinite set of values). The CDF of a discrete RV is piecewise linear. The CDF of a continuous RV is everywhere differentiable.

Continuing the example, consider how use the CDF to calculate the probability of a dart hitting the center third of the dartboard.

Given a set of observations of a random variable, one estimates the shape of the RV's CDF using the Empirical Distribution Function (EDF), which is a plot analagous to the CDF, but constructed from the values of the observations.

2.5 Probability Density Functions

Consider a sigmoid shaped CDF. What does it mean that the slope is steep in the middle? Answer: lots of values are probable there.

The slope tells us how likely values are in a particular range. It's important enough that we use it very often; it's called the *probability density function* (PDF). By definition, the PDF is the derivative of the CDF:

$$f(x) = \frac{dF(x)}{dx}$$

So note that the CDF can be expressed in terms of the PDF:

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

You should be able to derive:

$$\int_{-\infty}^{+\infty} f(x)dx = 1$$

$$f(x) \geq 0$$

What does the absolute value of the PDF mean? Nothing but the slope of the CDF. Do not interpret it as a probability.

Now, if an RV is discrete, its CDF is nondifferentiable. Therefore we need a different definition.

2.5.1 PDFs of discrete RVs

For the PDF of discrete RVs, we simply plot the probability function of each value. That is, we plot $P[X = x]$ for the various values of x.

Another way to think of the PDF is that it consists of impulses at the points of discontinuity of the CDF.

As an example, plot the PDF and CDF of the roll of two dice.

Be sure you understand the distinction between the discrete PDF and the continuous PDF. They are different!

2.6 Expected Value

Definition: The *expected value* $E[X]$ of a random variable X is the weighted sum (integral) of all possible values of the R.V. For a discrete random variable, this is:

$$E[K] \equiv \sum_{-\infty}^{+\infty} kP[K = k] \tag{2.1}$$

and for a continuous random variable with pdf p():

$$E[K] \equiv \int_{-\infty}^{+\infty} kp(k)dk \tag{2.2}$$

The expected value is also called the average or the mean, although we prefer to reserve those terms for empirical statistics (actual measurements, not idealizations like these formulas). The expected value is in some sense the “center of mass” of the random variable. It is often denote μ.

μ is an idealized notion, like the random variable itself. If we are dealing with a real dataset, all we can do is *estimate* what μ might be. For this purpose we use the *empirical mean* or *average,* denoted $\bar{x}$, which is (of course):

$$\bar{x} = 1/n \sum_{i=1}^{n} x_i$$

for observations $x_i, i = 1..n$.

The mean may not be very informative, or important. In some cases a random variable may not ever even take on the mean as a possible value. In other cases the notion of average isn't useful, as for the person with their head in the oven and feet in the freezer who claims "on average I feel fine." The mean may not be very important when observations are highly variable.

2.7 Variance

In fact, the variability of random processes is a central issue in performance evaluation. To describe variability we most commonly use *variance.* Variance is the average squared difference of the random variable from its mean. It is defined as:

$$Var(X) \equiv E[(X - E[X])^2]$$

so for a discrete R.V. with $E[X] = \mu$ this would be:

$$Var(X) = \sum_{x=-\infty}^{+\infty} (x - \mu)^2 P[X = x].$$

The units of variance are the square of the units of the mean. To compare variance and mean, we take the square root of the variance. This is called the *standard deviation* and is denoted σ. So variance is denoted σ^2.

We can form a good estimate of $Var(X)$ for a sample dataset using this formula:

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

And sample standard deviation (S) is the positive square root of S^2.

A crucial point is that the importance of variability depends on the magnitude of typical values. If I tell you that a particular species of animal typically varies over a range of plus or minus 10 pounds this is a lot more surprising if the animal weighs 15 pounds on average than if it weighs 150 pounds on average. That is, in the first case some individuals are five times larger than others (25 vs 5) while in the second case, we would consider most animals to be about the same in size.

Thus, we often need to normalize the standard deviation by the mean. This is called the Coefficient of Variation (cv) and is defined as σ/μ. Thus is someone tells you that a RV has a cv of 1/5, you know that the variability is small, while if the cv is 5, variability is large.

2.8 Probability Laws.

2.8.1 Law of total probability.

Let $A_1, A_2, ..., A_n$ be events such that

1. $A_i \cap A_j = \emptyset$ if $i \neq j$
2. $P[A_i] > 0, i = 1, 2, ..., n.$
3. $A_1 \cup A_2 \cup ... \cup A_n = \Omega$

(Such a family of events is called a *partition* of Ω.)

Then for any event A,

$$P[A] = P[A_1]\, P[A|A_1] + P[A_2]\, P[A|A_2] + ... + P[A_n]\, P[A|A_n]$$

2.8.2 Law of total expectation.

Conditional expectation is defined as follows:

$$E[X|Y = y_j] = \sum_{x_i} x_i\, P[X = x_i|Y = y_j]$$

(Note that this definition follows directly from the definition of expectation we have already seen.) The the law of total expectation states:

$$E[X] = \sum_{y_i} E[X|Y = y_i]\, P[Y = y_i]$$

Note that the set y_i forms a partition over the possible values of Y.

2.8.3 Bayes' Rule.

Suppose that $A_1, A_2, ..., A_n$ form a partition of Ω. Then for any event B with $P[B] > 0$,

$$P[A_i|B] = \frac{P[B|A_i]\, P[A_i]}{P[A_1]\, P[B|A_1] + P[A_2]\, P[B|A_2] + ... + P[A_n]\, P[B|A_n]}$$

You should be able to see how to derive this; steps are 1) expand into a fraction using the definition of conditional probability; 2) denominator then follows from law of total probability; 3) numerator is based on rewriting $P[A_i \& B]$ using definition of conditional probability again.

Note how we are updating our estimate of the probability of each A_i based on the new information, namely, that B is true. This update transforms the *prior* probabilities $P[A_i]$ into the *posterior* probabilities $P[A_i|B]$.

This is useful because often the probabilities $P[B|A_i]$ can be estimated.

Example. Empirical evidence suggests that amongs sets of twins, about 1/3 are identical. Assume therefore that probability of a pair of twins being identical to be 1/3. Now, consider how a couple might update this probability after they get an ultrasound that shows that the twins are of the same gender. What is their new estimate of the probability that their twins are identical?

Let I be the event that the twins are identical. Let G be the event that gender is the same via ultrasound. The prior probabilities here are $P[I]$ and $P[\bar{I}]$. What we want to calculate are the posterior probabilities $P[I|G]$ and $P[\bar{I}|G]$.

First, we note:

$$P[G|I] = 1$$

(Surprisingly, people are sometimes confused about that fact!) Also, we assume that if the twins are not identical, they are like any two siblings, *i.e.*, their probability of being same gender is 1/2:

$$P[G|\bar{I}] = 1/2$$

And we know from observing the population at large that among all sets of twins, about 1/3 are identical:

$$P[I] = 1/3$$

Then:

$$P[I|G] = \frac{P[G|I]P[I]}{P[G|I]\,P[I] + P[G|\bar{I}]\,P[\bar{I}]} = \frac{1 \cdot 1/3}{(1 \cdot 1/3) + (1/2 \cdot 2/3)} = \frac{1}{2}$$

So we have updated our estimate of the twins being identical from 1/3 (prior probability) to 1/2 (posterior probability).

Note how easy it was to determine $P[G|I]$ and $P[G|\bar{I}]$, while the other case $P[I|G]$ was not so obvious – hence the utility of Bayes' rule.

Exercises

Reasoning about Models

2-1. For two independent events A and B, $P[A] = 0.2$ and $P[B]$ = 0.3. What is the probability $P[A \cup B]$?

2-2. One of the most common assumptions made in performance analysis is that two events (or more) are independent.

(Rozanov) Let $A1$ be the event that a card picked at random from a full deck is a spade, and $A2$ be the event that it is a queen. Are $A1$ and $A2$ independent events?

2-3. Marks on the number line.

(a) Draw the number line from 1 to 10. For each integer position, I choose to place a mark with probability p (by flipping a coin which comes up heads with probability p). Now: sum the marks from 1 to 5, and from 6 to 10. Are the two sums independent?

(b) Draw the same number line. Now pick an integer at random from 1 to 10. Place a mark there. Do this (in an independent manner) N times. Now construct the same two sums as before. Are the two sums independent?

2-4. (Rozanov) In throwing a pair of dice, let $A1$ be the event that the first die turns up odd, $A2$ the event that the second die turns up odd, and $A3$ the event that the total number of spots is odd. Are $A1$ and $A2$ independent? Are $A1$ and $A3$ independent? Are $A1, A2$, and $A3$ all independent?

2-5. Consider two mutually exclusive events, A and B, each with positive probability (*i.e.,* $P[A \cup B] = P[A] + P[B]$ and $P[A] > 0, P[B] > 0$). Are A and B independent? Why or why not?

2-6. (Rozanov) A motorist encounters four consecutive traffic lights, each equally likely to be red or green (and these traffic lights can *only* be red or green). Let ξ be the number of consecutive green lights passed by the motorist before being stopped by a red light or passing through all lights. What is the probability distribution of ξ, expressed as a formula?

2-7. Prove that if $P[A|B] > P[A]$, then $P[B|A] > P[B]$.

Reasoning about Data

2-8. Consider the following situations. Analyze for independence.

At the campus trolley across from Warren towers. Consider the sample space the set of items that a customer might order (felafel, hummus, tabooli, fajita).

- Two customers are chosen at random. Is is reasonable to model their orders as being independent?
- Two customers arrive one after the other. Is it reasonable to treat their orders as independent?
- A customer orders two items. Is it reasonable to assume that the first item ordered is independent of the second item?
- A customer orders an item at 10am. Is reasonable to assume the time the order is made independent of the item ordered?
- A customer orders items on consecutive days. Is it reasonable to assume the orders independent?

Working With Data

2-9. You are to write four programs that you will use in subsequent assignments.

(a) `mean` - calculate the mean of a dataset

(b) `variance` - calculate the variance of a dataset

(c) `hist <nbins>` - calculate histogram of a dataset using `nbins` bins.

(d) `cdf` - calculate the CDF of a dataset.

Each program should accept an arbitrary-sized dataset. It should assume all datasets consist of real numbers separated by linefeeds. The programs `hist` and `cdf` should output data in a form that can immediately be plotted (*e.g.,* by `gnuplot`). This means that the program outputs should be *x, y* pairs separated by linefeeds.

In producing a histogram, one must construct some number of equal-sized bins, and then count how many observations fall in each bin. To do this, calculate the max and min of the dataset, and divide the difference into `nbins-1` equal sized bins with cutoff values x_i, $i = 1...nbins - 1$. For each bin bounded by x_i and x_{i+1}, count the number of data points x such that $x_i \leq x < x_{i+1}$. Finally you will have to deal with the "fencepost" problem: there will be a number of values not allocated to any bin because they are the maximum values in the dataset. Create one more bin from x_{nbins} to $x_{nbins+1}$ to hold these values.

Also note that if the dataset consists of discrete values (like integers) then you should make sure your bins each include the same number of discrete values (like, 1 value per bin) and the bins should be suitably labeled.

In producing the CDF, you should not simply sum up the histogram, because (for a continuous random variable) the histogram is only an approximation to the PDF. Instead, make sure that each distinct value in the input dataset results in a distinct point on the CDF. That is, for the case of a continuous-valued dataset, the CDF should be a smooth curve.

There are three datasets on the course web page named `ds1, ds2, ds3.` Run all four programs on the datasets and collect the output. Also calculate the coefficient of variation for each dataset. For the histogram, you must decide (by inspecting the results) on the proper number of bins to make the output as meaningful as possible.

Chapter 3

Important Distributions

3.1 The “Independent Events” Distributions

Canonical experiment: flipping a weighted coin; coin comes up “heads” (success) with probability p. Fundamental assumption is *independence*. These distributions are about independent events.

	Trials or Time until Success	Number of Success in Fixed Time or # Trials
Discrete Trials	Geometric	Binomial
Continuous Rate	Exponential	Poisson

3.2 Discrete Distributions

3.2.1 Geometric

Experiment: flip a weighted coin until first success. How many flips does it take to get a success?

Note: $q = 1 - p$.

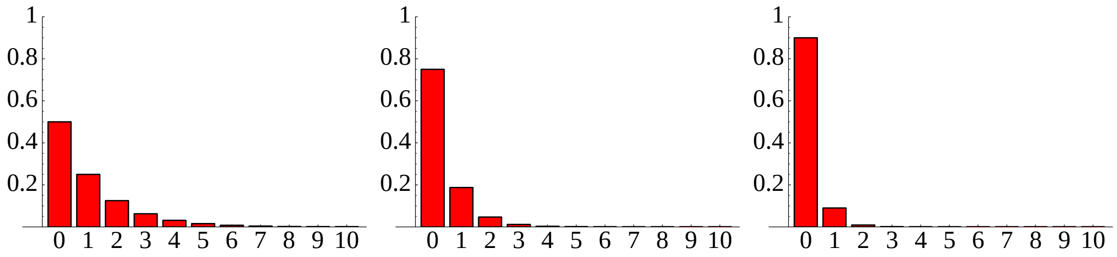
$$\begin{aligned} P[X = k] &= p(1-p)^{k-1} = pq^{k-1} \\ F(k) &= \sum_{n=1}^{k} P[X = n] \\ &= \sum_{n=1}^{k} p(1-p)^{n-1} \\ &= \sum_{n=0}^{k-1} pq^{n} \end{aligned}$$

$$\begin{aligned} &= p\sum_{n=0}^{k-1} q^n \\ &= p\,\frac{1-q^k}{1-q} \\ &= 1-q^k \end{aligned}$$

The mean of this distribution is $1/p$. Note that the mean number of failures before the first success is 1 minus this value, *i.e.*, $1/p - 1 = 1 - p/p$. Also note that sometimes we will work with p (prob. of success) and sometimes with q (prob. of failure); for example, the mean number of failures before a success, in terms of the probability of failure, is $q/1 - q$.

Variance: $(1-p)/p^2$

C.V. = $1 - p$



$p = 0.5$ $p = 0.75$ $p = 0.9$

Figure 3.1: Geometric Distributions

3.2.2 Binomial

Experiment: precisely N coin flips; k is the number of successes. p is the probability of a success.

P[any particular sequence of N trials with k successes] $= (p)^k(1-p)^{N-k}$.

But there are many different such sequences: C_k^N of them in fact.

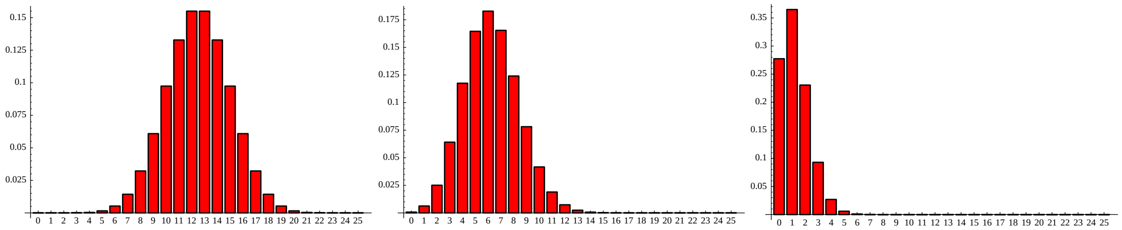
So $P[\#\text{ successes } = k] = C_k^N(p)^k(1-p)^{N-k}$

CDF. $F(k) = \sum_{n=0}^{k} P[\#\text{ successes } = n]$. Note: no closed form.

Mean = pN

Variance = $Np(1-p)$

CV is less than 1



$p = 0.5$ $p = 0.25$ $p = 0.05$

Figure 3.2: Binomial Distributions with $N = 25$

3.2.3 Poisson

Limiting form of binomial, when the number of trials goes to infinity, happening at some rate λ.

$$P[k \text{ successes in time } T] = (\lambda T)^k \frac{e^{-\lambda T}}{k!}$$

CDF: no closed form

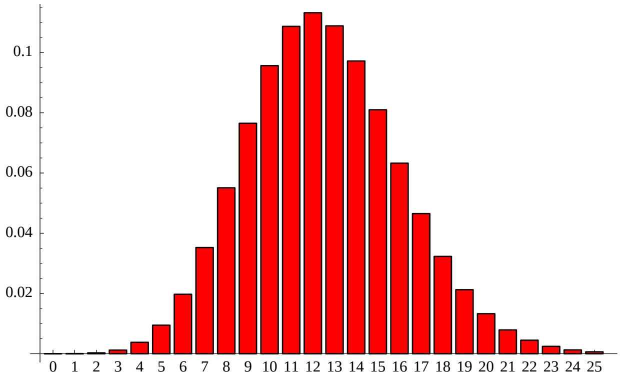


Figure 3.3: Poisson Distribution with $\lambda T = 12.5$

mean: λT

variance: λT

cv : $1/\sqrt{\lambda T}$ (ie, < 1)

3.3 Continuous Distributions.

3.3.1 Uniform

On some range (a, b), the Uniform distribution simply states that all possibilities are equally likely. So:

$$p(x) = 1/(b-a) \quad a \leq x \leq b$$

Exercise: Compute the mean, variance, and CV of this distribution as a function of a and b.

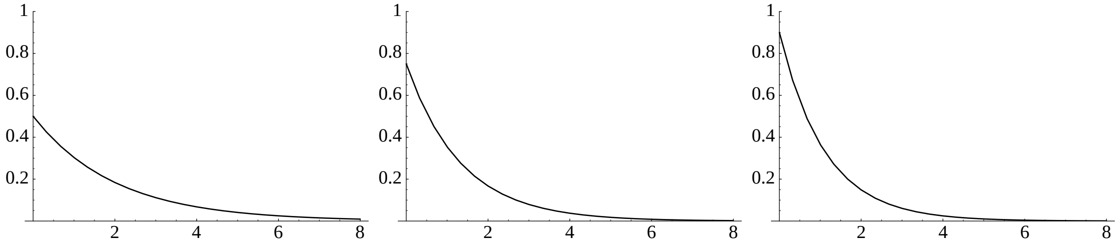
3.3.2 Exponential

This distribution is the analog of the geometric in the continuous case, *i.e.*, the situation in which a success happens at some rate λ. This distribution measures the time until a success occurs.

$$\begin{aligned}
p(x) &= \lambda e^{-\lambda x} \\
F(x) &= \int_0^x \lambda e^{-\lambda y}\, dy \\
&= -e^{-\lambda y}|_0^x \\
&= 1 - e^{-\lambda x}
\end{aligned}$$

$$\begin{aligned}
E[X] &= \int_0^\infty x\, \lambda e^{-\lambda x}\, dx \\
&= -\lambda \int_0^\infty (\frac{d}{d\lambda} e^{-\lambda x})\, dx \\
&= -\lambda \frac{d}{d\lambda} \int_0^\infty e^{-\lambda x}\, dx \\
&= -\lambda \frac{d}{d\lambda} \left(-\frac{1}{\lambda} e^{-\lambda x}\right) |_0^\infty \\
&= \left(-x - \frac{1}{\lambda}\right) e^{-\lambda x}|_0^\infty \\
&= \frac{1}{\lambda}
\end{aligned}$$

(Can also solve the expectation using integration by parts, with $u = -e^{-\lambda x}$ and $v = x$, so $du = \lambda e^{-\lambda x}\, dx$ and $dv = dx$, then following the rule that $\int v\, du = uv - \int u\, dv$.)



$\lambda = 0.5$ $\lambda = 0.75$ $\lambda = 0.9$

Figure 3.4: Exponental Distributions

So mean: $1/\lambda$

Likewise, variance: $1/\lambda^2$

So cv: 1

3.3.3 Normal.

Important properties of the normal distribution:

1. The sum of n independent normal variates is a normal variate. Thus normal processes remain normal after passing through linear systems. ($X_1 = aX_2 + bX_3$ is normal if X_2, X_3 are.)

2. The sum of a large number of independent observations from any distribution with finite variance tends to have a normal distribution. This is the Central Limit theorem.

Thus we can see that one way of thinking of the Normal is that it is the limit of the Binomial when n and p are large, that is, the limit of the sum of many Bernoulli trials. However many other sums of random variables (not just Bernoulli trials) converge to the Normal as well.

Normal is widely used; often asserted to be most common distribution - but it's not so in computer systems. Gaussian distributions have extremely "light" tails.

For example, under a Gaussian model, the typical deviation of x from μ is σ. However, the respective probabilities of that x deviates from μ by more than 2σ, 3σ, 4σ, and 5σ are 0.046, 0.003, 6 $\times 10^{-5}$, and 6 $\times 10^{-7}$. Thus deviations from the mean of 5σ should not be seen in practice; but this is actually not uncommon in computer systems data.

From [Mac03, p.312]:

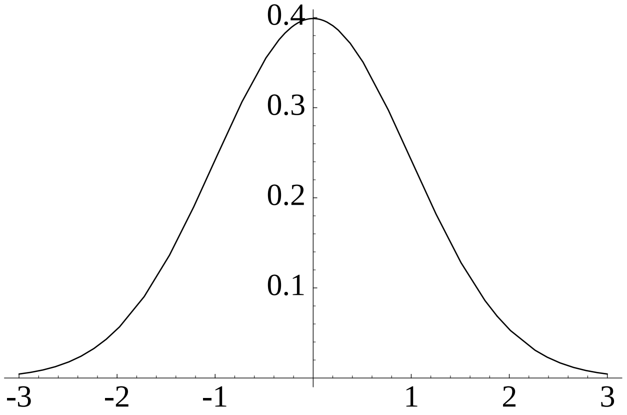


Figure 3.5: Normal Distribution with $\mu = 0, \sigma = 1$

> ... if a variable that is modelled with a Gaussian actually has a heavier-tailed distribution, the rest of the model will contort itself to reduce the deviation of the outliers, like a sheet of paper being crushed by a rubber band.

3.4 Heavy-Tailed Distributions

The "light-tail" assumption is often not true for computer systems.

A particularly important part of a distribution is the shape of its upper tail. Tail shape determines the likelihood of large observations, which in the case of network measurements can often dominate system performance.

Many of the most commonly used distributions have tails which decline exponentially or faster. The Normal distribution, the Exponential distribution, and the Uniform distribution all fall in this category. For such distributions, the likelihood of an extremely large observation (*e.g.,* many standard deviations above the mean) is negligible or zero.

Supplemental Material: In contrast, a distribution whose upper tail declines more slowly than any exponential is referred to as a *subexponential* distribution [GK98]. This distributions have very high, or even infinite, variance. The practical result is that samples from such distributions show extremely large observations with non-negligible frequency. Formally we define a subexponential distribution as one for which

$$(1 - F(x))e^{ax} \to \infty \text{ as } x \to \infty \text{ for all } a > 0.$$

A special case of the long-tailed (or subexponential) distribution is the *heavy-tailed* distribution. Such distributions have tails that asymptotically approach a hyperbolic (power-law) shape. Formally these are distributions for which

$$1 - F(x) \sim x^{-\alpha}$$

where $a(x) \sim b(x)$ means that $lim_{x \to \infty} a(x)/b(x) \to c$ for some constant c.

Heavy tailed distributions behave quite differently from the distributions more commonly used in performance evaluation (*e.g.,* the Exponential). In particular, when sampling random variables that follow heavy tailed distributions, the probability of very large observations occurring is non-negligible. In fact, under our definition, heavy tailed distributions have *infinite variance,* reflecting the extremely high variability that they capture; and when $\alpha \leq 1$, these distributions have *infinite mean.*

Thus the fact that a distribution is subexponential or heavy-tailed depends only on the shape of the upper tail and not on the body of the distribution.

Note that subexponentiality and heavy tails are properties of models, not data. However a given dataset may be consistent with a subexponential or heavy tailed distribution, meaning that for practical purposes we get good answers to engineering questions if we use such distributions as a model for the data. This phenomenon in data is known as *long tails.* The phenomenon is often generically called “heavy tails” but we will reserve that term specifically for distributions (having power-law tail shape).

Subexponential distributions and heavy tails are reviewed in [AFT98].

There are many examples of long-tailed data found in computer systems.

The evidence for heavy-tailed distributions in a number of aspects of computer systems is now quite strong. The broadest evidence concerns the *sizes of data objects* stored in and transferred through computer systems; in particular, there is evidence for heavy tails in the sizes of:

- Files stored on Web servers [AW97, CB97];
- Data files transferred through the Internet [CB97, Pax94];
- Files stored in general-purpose Unix filesystems [Irl94]; and
- I/O traces of filesystem, disk, and tape activity [GMR^{+}98, PG95, Pet96, PA96]

Next, measurements of *job service times or process execution times* in general-purpose computing environments have been found to exhibit heavy tails [GLR92, HBD97, LO86].

A third area in which heavy tails have recently been noted is in the distribution of *node degree of certain graph structures*. Faloutsos *et al.* [FFF99] show that the inter-domain structure of the Internet, considered as a directed graph, shows a heavy-tailed distribution in the outdegree of nodes. Another study shows that the same is true (with respect to both indegree and outdegree) for certain sets of World Wide Web pages which form a graph due to their hyperlinked structure [AJB99]; this result has been extended to the Web as a whole in [BKM$^+$00].

In practice, random variables that follow heavy tailed distributions are characterized as exhibiting many small observations mixed in with a few large observations. In such datasets, most of the observations are small, but most of the contribution to the sample mean or variance comes from the rare, large observations. This means that those sample statistics that are defined converge very slowly. This is particularly problematic for simulations involving heavy tails, which many be very slow to reach steady state [CL99].

3.4.1 Zipf's Law.

Finally, a phenomenon related to heavy tails is the so-called *Zipf's Law* [Zip49]. Zipf's Law relates the "popularity" of an object to its location in a list sorted by popularity. More precisely, consider a set of objects (such as Web servers, or Web pages) to which repeated references are made. Over some time interval, count the number of references made to each object, denoted by P. Now sort the objects in order of decreasing number of references made and let an object's place on this list be denoted by r. Then Zipf's Law states that

$$P = cr^{-\alpha}$$

for some positive constants c and α. In its original formulation, Zipf's Law set $\alpha = 1$ so that popularity (P) and rank (r) are inversely proportional. In practice, various values of α are found, with values often near to or less than 1. Evidence for Zipf's Law in computing systems (especially the Internet) is widespread [ABCdO96, CBC95, Gla94, NHM$^+$98]; a good overview of such results is presented in [BCF$^+$99].

Thus Zipf's Law is an analog of the long tails phenomenon occuring in categorical (non-numeric) data. One can form an analog of the histogram for categorical data by counting the number of occurrences of items in each category, and sorting the result in decreasing order.

Note the relationship between Zipf's law and an actual distribution. In Zipf's Law, if you rank the objects by popularity (or rate) then the popularity $x_{(r)}$ as a function of rank r is approximately $rx_{(r)} = constant$ or more generally $r^{\alpha}x_{(r)} = constant, \alpha > 0$.

This then can be used as a distribution by normalizing to unit sum. That is, assuming there are N possible values:

$$p(x_{(r)}) = \frac{r^{-\alpha}}{\sum_{i=1,\ldots,N} i^{-\alpha}}$$

This is called the *Zipf distribution.*

Since high variance (indeed, long tails) is a common characteristic of computing systems data, we often some additional distributions (other than the Zipf) that can be used to model such data.

3.4.2 Pareto

The Pareto distribution is the simplest continuous heavy-tailed distribution. It is the continuous analog of the Zipf distribution. It declines hyperbolically (*i.e.*, polynomially). It is given by:

$$p(x) = \alpha k^\alpha x^{-\alpha-1} \quad k \leq x, 0 < \alpha \leq 2.$$

It takes on values in the range $[k, \infty]$. Note that if $\alpha > 2$, this is still a distribution, but it is not heavy-tailed and generally not referred to as a "Pareto" distribution.

Pareto was an Italian economist who studied income distributions. (In fact, income distributions typically show heavy tails.)

Its CDF is particularly simple:

$$F(x) = 1 - (k/x)^\alpha \quad k \leq x, 0 < \alpha \leq 2.$$

3.5 Coxian Distributions

A number of distributions can be "constructed" from exponentials. These are useful because often it is particularly easy to work with exponentials. So, for example, rather than using a polynomially-tailed distribution, it may be an acceptable model to use a hyperexponential.

3.5.1 Hyperexponential

The Pareto is often a good fit to computer systems data , but it can sometimes be hard to work with in practice. A distribution that is easier to work with, and yet can be constructed with arbitrarily large variance is the *hyperexponential.* It takes three parameters: $\mu_1 > 0$, $\mu_2 > 0$, and $0 < q < 1$.

is given by:

$$p(x) = q\mu_1 e^{\mu_1 x} + (1-q)\mu_2 e^{-\mu_2 x}$$

Thus it is a *mixture* of two exponentials. It can be though of as follows: flip a coin with probability of heads $= q$. If the coin comes up heads, take a sample of an exponential distribution with parameter μ_1; otherwise, take sample of an exponential distribution with parameger μ_2.

The variance of the hyperexponential is:

$$\sigma^2 = \frac{2q}{\mu_1^2} + \frac{2(1-q)}{\mu_2^2} - \left(\frac{q}{\mu_1} + \frac{1-q}{\mu_2}\right)^2$$

Since there are three free parameters, there are many ways to get a distribution with a given mean and variance. One way is as follows:

Given $1/\mu$ (the desired mean) and C^2 (the desired squared coefficient of variation, i.e., $\sigma^2\mu^2$, let:

$$\begin{aligned} q &= \frac{1}{2}\left(1 - \left[\frac{C^2-1}{C^2+1}\right]^{\frac{1}{2}}\right) \\ \mu_1 &= 2q\mu \\ \mu_2 &= 2(1-q)\mu \end{aligned}$$

This particular choice of q, μ_1, μ_2 has the property of "balanced means":

$$\frac{q}{\mu_1} = \frac{1-q}{\mu_2}$$

Mixtures of (larger numbers of) exponentials can in fact be used to approximate *any* distribution function. In particular, for approximating a Pareto using mixtures of exponentials see [FW97].

3.5.2 Erlang-k

There are also situations in which we are interested in low-variance distributions. One useful distribution in this regard is the Erlang-k. An Erlang-k random variable is the sum of k exponential random variables each with parameter λ.

As one sums random variables, eventually the Central Limit Theorem takes hold. Thus for large k, the Erlang-k is approximately normal, while for $k = 1$ it is exponential.

A. K. Erlang was a Danish mathematician who contributed immensely to the design of early telephone networks and switching systems.

It has distribution function:

$$p(x) = \frac{\lambda k (\lambda k x)^{k-1}}{(k-1)!} e^{-\lambda k x} \quad x > 0.$$

One can think of the Erlang-k distribution as the time required by a customer to pass through k service centers, each of which requires time given by an exponential distribution with rate λk.

The mean of this distribution is $1/\lambda$. Thus it has the same mean as an exponential distribution with rate λ.

The variance of this distribution is:

$$\sigma^2 = \frac{1}{k\lambda^2}$$

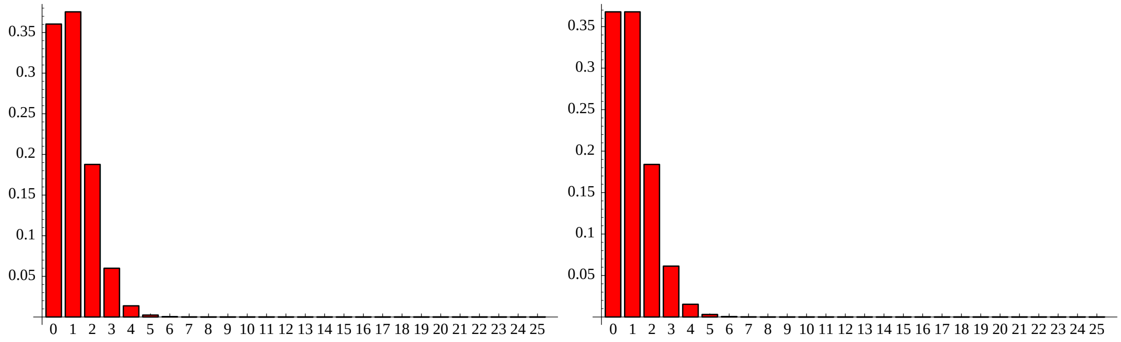
.

As can be seen, for a given mean, the variance of the Erlang-k decreases in inverse proportion to k.

3.6 Approximations

If n is large and p (= prob success) is small, the poisson is a good approximation for the binomial.

Figure 3.6 shows this. It shows the Binomial with large n and small p compared to the Poisson with same mean (1). Note how similar the figures look.



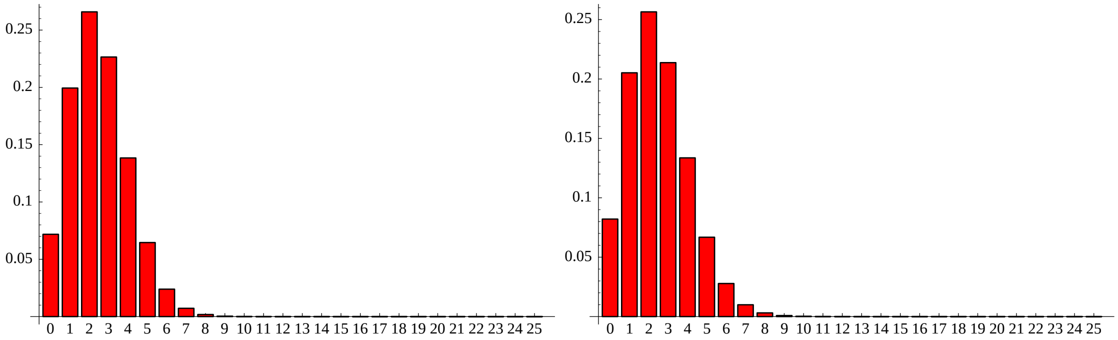
Binomial, $n = 25, p = 0.04$

Poisson,$\lambda T = 1$

Figure 3.6: Comparing Binomial and Poisson with Small Mean ($E[X] = 1$), Large No. of Trials

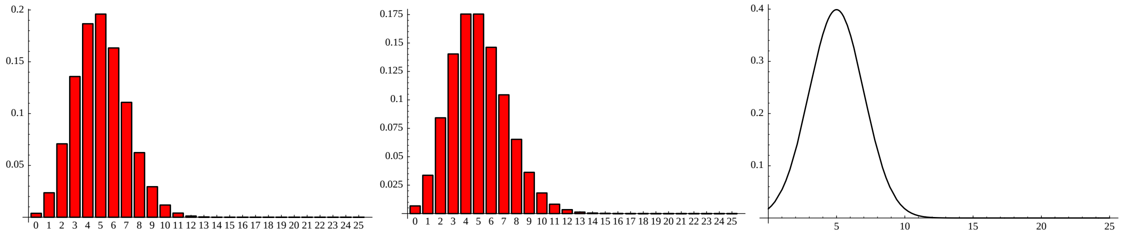
Figure 3.7 shows how both distributions change as we increase the mean to 2.5. Note how the figures both become more symmetric.

In fact, eventually for large mean, they become completely symmetric, and well approximated by the Normal distribution! Figure 3.8 shows the same two distributions with mean of 5, alongside the Normal with same mean and variance. In the limit of large number of trials and large mean, both Poisson and Binomial are well approximated by Normal.



Binomial, $n = 25, p = 0.1$ Poisson, $\lambda T = 2.5$

Figure 3.7: Comparing Binomial and Poisson with Larger Mean ($E[X] = 2.5$), Large No. of Trials

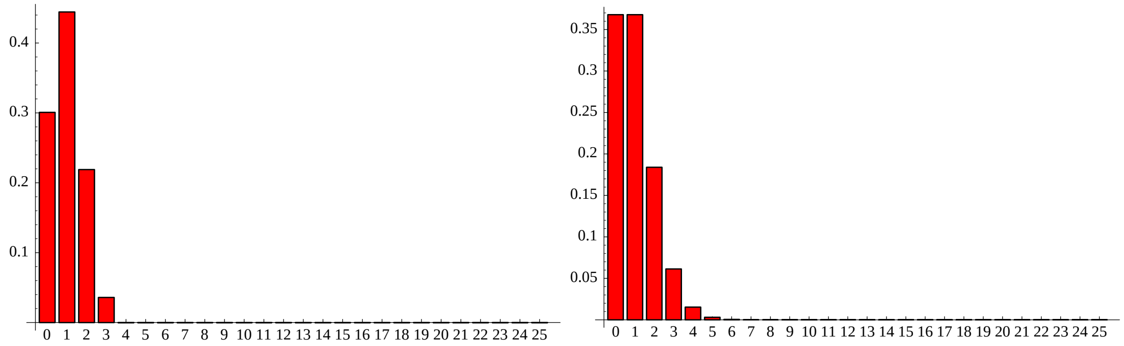


Binomial, $n = 25, p = 0.2$ Poisson, $\lambda T = 5$ Normal, $\mu = 5, \sigma = 2.$

Figure 3.8: Comparing Binomial, Poisson, and Normal with Large Mean ($E[X] = 5$), Large No. of Trials

The variance of the Binomial is $np(1-p)$, so for these values that yields a standard deviation of 2, which is what is used to find the matching Normal distribution.

However, if n is small, one cannot do good approximations using the Poisson or the Normal; in that case, one must use the Binomial itself. See Figure 3.9 for a comparison of Poisson and Binomial in the case where $n = 3$. Note how bad the match is.



Binomial, $n = 3, p = 0.33$

Poisson, $\lambda T = 1$

Figure 3.9: Comparing Binomial and Poisson with Same Mean, Small No. of Trials

Exercises

Reasoning about Models

3-1. Prove the Law of Total Expectation.

3-2. If X and Y are independent random variables, prove that $E[XY] = E[X]\,E[Y]$.

3-3. Which of these distributions would you use to describe ...

- the number of processors that are up in a multiprocessor system?
- the number of bits successfully transmitted in a row?
- number of requests to a server in a given time interval T?
- the number of local requests to a Web page between remote requests to a Web page?
- the number of packets that reach a destination without loss?
- the number of queries to a database in S seconds?
- the number of bits in a packet that are unaffected by noise?
- the number of processor failures in an hour?

3-4. Given a random variable X that is exponentially distributed with parameter λ. What is the probability that any particular observation of X will be greater than its mean?

3-5. (Allen) A certain airline has found that approximately 5% of all persons holding reservations on a given flight do not show up. The plane holds 50 passengers and the airline takes reservations for 53 (this is called *overbooking*). What is the probability that every passenger who arrives on time for the flight will have a seat? (Assume there are no walk-ins.)

3-6. (Allen) Personnel of a certain engineering company use an online terminal to make routine engineering calculations. If the time each engineer spends in a session at a terminal has an exponential distribution with an average value of 36 minutes, find

(a) The probability that an engineer will spend 30 minutes or less at the terminal,

(b) The probability that an engineer will use it for more than an hour.

(c) If an engineer has already been at the terminal for 30 minutes, what is the probability that he or she will spend more than another hour at the terminal?

(d) Ninety percent of the sessions end in no more than R minutes. What is R?

3-7. A programmer submits jobs to a system as follows: she first submits a job of size 1. With probability $1/2$, she is satisfied with the result and stops there; however with probability $1/2$ she is not satisfied, and proceeds to submit a job of size 2. The process then repeats; with probability 1/2 she stops, and with probability 1/2 she goes on to submit a job of size 4, and so on. Once the whole process completes, she starts over again with a job of size 1. Let the random variable X be the size of a job.

(a) Is X discrete or continuous?

(b) What are the possible values X may take on?

(c) What is the probability distribution of X?

(d) What is the expected size of a job?

3-8. A *Bernoulli* random variable is one that takes on only the values 0 or 1. Consider a Bernoulli random variable B in which the probability of taking on the value 1 is p.

(a) What is the expected value of B?

(b) What is the variance of B?

(c) For what value(s) of p is the variance of B maximized?

(d) For what value(s) of p is the variance of B minimized?

Working with Models

3-9. On a single plot, show:

- An exponential pdf with $\mu_1 = 1$
- An exponential pdf with $\mu_2 = 0.2$
- A hyperexponential pdf with $q = 0.5, \mu_1 = 1, \mu_2 = 0.2$.

3-10.

- On a single plot, show:
  - A pdf (histogram) of the Binomial distribution with $N = 40, p = 0.6$
  - A pdf (continuous curve) of a Normal distribution that you have chosen as a good approximation to this Binomial.
- State the parameters of the Normal that you used and how you got them.
- State the parameter(s) of the Poisson distribution you would use to approximate this Binomial.

Chapter 4

Exponentials and the Poisson Process

4.1 Stochastic Processes

A *stochastic process* $\{X(t), t \in T\}$ is a collection of random variables. That is, for each $t \in T$, $X(t)$ is a random variable. The index t is often interpreted as time, and we refer to $X(t)$ as the state of the process at time t.

The set T is the index set of the process. When T is a countable set the stochastic process is said to be a *discrete-time* process. When T is an interval of the real line, the stochastic process is said to be a *continuous time* process.

So for example we can have the discrete time process $\{X_n, n = 0, 1, ...\}$ or the continuous time process $\{X(t), t \geq 0\}$.

An equivalent view of a stochastic process is a set of *realizations:* $\{x_1, x_2, ...\}$; each realization occurs with probability equal to the joint distribution of all individual observations. The set of all realizations forms an *ensemble.*

The RVs themselves ($X(t)$) may be continuous valued or discrete valued. These are called *continuous state space* and *discrete state space* systems (or chains).

Relationship between counting process and interarrival process.

Still to cover: ergodicity, stationarity.

4.2 Memoryless Property of the Geometric Distribution.

Consider the canonical coin-flipping experiment, in which we are counting the number of trials up to and including the first success. Suppose we are given that there were no successes during the first m trials. We wish to calculate the probability that there will be k more trials up to and including the first success, that is, $P[T = k + m|T > m]$ for $k = 1, 2, ...$

By the definition of conditional probability, we have:

$$P[T = k + m|T > m] = \frac{P[(T = k + m)\&(T > m)]}{P[T > m]}$$

but

$$\{T = k + m\}\&\{T > m\} = \{T = k + m\}$$

since $k > 0$, and

$$P[T > m] = 1 - P[T \leq m] = q^m$$

so

$$P[T = k + m|T > m] = \frac{pq^{m+k-1}}{q^m} = pq^{k-1} = P[T = k]$$

This is called the Markov or memoryless property of the geometric distribution. This should make sense to you if you think about coin flipping as the canonical experiment for the geometric distribution ... the coin doesn't know what happened in the past.

4.3 Properties of the Exponential RV

4.3.1 Comparing Two Exponential Random Variables

Suppose we have two independent random variables X_1 and X_2 that are each exponentially distributed, with respective parameters λ_1 and λ_2. Let us determine the probability that X_1 is less than X_2.

We can attack this problem by conditioning on X_1:

$$\begin{aligned} P[X_1 < X_2] &= \int_0^\infty P[X_1 < X_2|X_1 = x]p(x)dx \\ &= \int_0^\infty P[X_1 < X_2|X_1 = x]\lambda_1 e^{-\lambda_1 x}dx \\ &= \int_0^\infty P[x < X_2]\lambda_1 e^{-\lambda_1 x}dx \end{aligned}$$

$$
\begin{aligned}
&= \int_0^\infty e^{-\lambda_2 x}\lambda_1 e^{-\lambda_1 x}dx \\
&= \int_0^\infty \lambda_1 e^{-(\lambda_1+\lambda_2)x}dx \\
&= \frac{\lambda_1}{-(\lambda_1+\lambda_2)}e^{-(\lambda_1+\lambda_2)x}|_0^\infty \\
&= \frac{\lambda_1}{-(\lambda_1+\lambda_2)}(0-1) \\
&= \frac{\lambda_1}{\lambda_1+\lambda_2}.
\end{aligned}
$$

4.3.2 Minimum of n Exponential Random Variables

Suppose that $X_1, X_2, ..., X_n$ are independent exponential random variables, with X_i having rate $\mu_i, i = 1, ..., n$. Now let us define a new random variable Y = minimum$(X_1, ..., X_n)$.

Make sure you are clear about what is meant here: Y is a random variable that takes whatever value happens to be the minimum when we sample each of the X_i once.

It turns out that Y is also an exponential random variable, with rate equal to the sum of the μ_i. This is shown as follows:

$$
\begin{aligned}
P[Y > x] &= P[\text{minimum}(X_1, ..., X_n) > x] \\
&= P[X_1 > x, X_2 > x, ..., X_n > x] \quad \text{(this is the key step; why is it valid?)} \\
&= \prod_{i=1}^{n} P[X_i > x] \quad \text{(by independence)} \\
&= \prod_{i=1}^{n} e^{-\mu_i x} \\
&= e^{-(\sum_{i=1}^{n}\mu_i)x}
\end{aligned}
$$

4.3.3 Poisson Processes

Consider again the canonical experiment which gives rise to the Exponential distribution: we are flipping a coin infinitely fast, with a probability of success such that the expected time until a success is $1/\lambda$. Then the rate of successes is λ.

Now let us assume that as soon as we get a success, we immidiately start the process all over again. Then we have a sequence of successes separated by exponentially distributed lengths of time. This is called a *Poisson process.*

We have already seen that the number of successes in time T of a Poisson process with rate λ is described by the Poisson distribution with parameter λT.

We can denote a Poisson process as $N(t)$ where N counts the number of successes that have occurred at time t. (Counting is assumed to start at time 0).

Superposition of Poisson Processes

Now let us consider two Poisson processes $N_1(t)$ having rate λ_1 and $N_2(t)$ having rate λ_2. Suppose we *superpose* the two processes, meaning: at each time t, count the total number of successes that have occurred in *both* N_1 and N_2. That is, $N_S(t) = N_1(t) + N_2(t)$.

In this superposed process, a success occurs whenever a success occurs in *either* N_1 or N_2. How can we describe the process $N_S(t)$?

Consider an arbitrary point in time, and ask: "what is the distribution of time until a success occurs in N_S?". Clearly, this time is equal to the *minimum* of the time to the next success in either N_1 or N_2. The time to next success in N_1 is distributed exponentially with parameter λ_1, and correspondingly for N_2. So by the minimum-of-exponentials rule above, the time to the next success in N_S is also distributed exponentially, and the parameter is $\lambda_S = \lambda_1 + \lambda_2$.

So a remarkable fact about Poisson processes is that the superposition of two Poisson processes is *also* a Poisson process, with rate equal to the sum of the rates of the two components.

Splitting of Poisson Processes

It can be shown as well that probabilistic *splitting* of a Poisson process yields Poisson processes. That is, given a Poisson process $N(t)$ having rate λ, suppose that each time an event occurs it is classified as either a type I or a type II event. Suppose further that each event is classified as a type I event with probability p and a type II event with probability $1 - p$ independently of all other events. Let $N_1(t)$ and $N_2(t)$ denote respectively the number of type I and and type II events occurring in $[0, t]$. Note that $N(t) = N_1(t) + N_2(t)$. Then $N_1(t)$ and $)N_2(t)$ are both Poisson processes having respective rates $p\lambda$ and $(1 - p)\lambda$. Furthermore, the two processes are independent.

4.4 Path to the Poisson Process

How valid is the Poisson process as an assumption for arrivals? In fact there are some valid reasons for assuming that arrivals can often be approximated as a Poisson process.

The first evidence is empirical. Lots of random processes have been shown to be well modeled by a Poisson process. For example, the number of

- Prussian cavalry officer deaths by horsekick

- flying-bomb hits on London in World War II
- accidents on a given stretch of road
- newspaper misprints on a page
- occurrences of the word ’and’ in President Bush’s state of the union speeches (see `http://www.nytimes.com/ref/washington/20070123_STATEOFUNION.html`)

are all well described as Poisson processes.

The general reason that this occurs is as follows. The limit of a large number of independent stationary renewal processes (each with arbitrary distribution of renewal time) will tend to a Poisson process.

That is, when the arrival process can be thought of as the superposition of a very large number of processes each occurring at a very low rate, the resulting process tends to a Poisson process.

When does this assumption fail? When finiteness of population is significant. That is, when the number of superimposed processes is not so large. In this case, the arrival of a customer can decrease the probability that a new customer will arrive. These are called *finite population* or *closed* queueing systems.

Poisson Arrivals See Time Averages (PASTA).

The “time average” state of a system means the average one sees if one chooses a time to observe at random, with uniform probability over some interval. This is distinct from the “arriving customer average” (*i.e.,* what an arriving customer typically sees) which could be very different from the time average. Consider the case in which customers arrive periodically with some fixed interarrival time. Then if the system also has some periodicity, it may happen that the arriving customer consistently sees the system in some “unusual” state. Or, for example, consider what happens when batches of customers arrive. The later-arriving customer *always* sees other customers in the system, even if this is not the typical state of the system.

So, what does observing the “time-average” mean? Pick any interval $[t, t+s]$ out of $[0, T]$. Then the probability of the observation falling in the interval should be s/T. We now show that the a Poisson process generates events uniformly over the interval $[0, T]$.

Divide the interval $[0, T]$ into k subintervals $h_1, h_2, ..., h_k$. We will show that the probability that exactly one Poisson point occurs in each of the subintervals is the same as the probability of uniformly selecting k points and finding exactly one in each of the subintervals.

The probability that exactly one of the k randomly selected points would falling each of the subintervals h_i is calculated as follows. The probability that a particular point hits a interval h_i is simply h_i/T and there are $k!$ ways this could happen, so:

$$P[\text{one point falls in each interval}] = \frac{k!}{T^k} h_1 h_2 ... h_k$$

Now, in the case of the Poisson process, we reason as follows. The Poisson process is memoryless, so each start of a subinterval can be considered a starting point for the Poisson process. This means that the probability of n points in any given interval is simply given by the Poisson distribution. What happens in each interval is independent, so the probability that precisely one Poisson arrival occurs in each interval is:

$$P[\text{one point in each interval}] \quad = \quad P[\text{one point in } h_1] \cdot P[\text{one point in } h_2] \cdot ... \; P[\text{one point in } h_k]$$

and the probability that precisely one Poisson arrival occurs in each interval given that k Poisson arrivals occurred is:

$$\begin{aligned} & P[\text{one point in each interval } | k \text{ points in total}] \\ = \; & \frac{P[\text{one point in } h_1] \cdot P[\text{one point in } h_2] \cdot ... \; P[\text{one point in } h_k]}{P[k \text{ points in } T]} \\ = \; & \frac{\lambda h_1 e^{-\lambda h_1} \, \lambda h_2 e^{-\lambda h_2} \, ... \, \lambda h_k e^{-\lambda h_k}}{\lambda^k e^{-\lambda T} T^k / k!} \\ = \; & \frac{k!}{T^k} h_1 h_2 ... h_k \end{aligned}$$

So we have the same probability assuming Poisson arrivals as we do when we assume random selection. Since we did not restrict the number of the intervals of their size, the result must be true for an arbitrary set of intervals positioned in an arbitrary way. So the location of those Poisson arrivals must be uniformly distributed in the interval $[0, T]$.

4.4.1 Timeseries; Dependence and Autocorrelation; SRD and LRD

TBD.

4.4.2 Mean Residual Life

Note: superseded by Palm Calculus approach of PE §11.

It often happens that we want to know the mean value of a random variable, given that we know the RV is greater than some value. This is a conditional mean, called *residual life.* For example: how long do I expect this lightbulb to burn, given that it has already been burning for 100 hours?

Now let us consider the mean residual life; that is, the average residual life, where the average is taken over all possible starting times. For example, for a set of a large number of lightbulbs that have all been burning for different (random) times, what is the average time that a bulb will keep burning? This is *mean residual life.*

We know how to form the probability distribution of X given that it is greater than some value:

$$P[X = k + s | X > k] = \frac{P[X = k + s]}{P[X > k]}$$

So to determine residual life after time k, we can write:

$$E[X|X > k] = \frac{\int_k^\infty xp(x)dx}{\int_k^\infty p(x)dx}$$

Then the mean residual life is the expectation of this over all k:

$$\bar{r} = E[E[X|X > k]] = \int_{-\infty}^{+\infty} p(k) \left(\frac{\int_k^\infty xp(x)dx}{\int_k^\infty p(x)dx} \right) dk$$

We can determine that this yields:

$$\bar{r} = \frac{E[X^2]}{2E[X]}$$

That is, the second moment over twice the first moment.

(not required material)
To see this, we reason as follows: Denote by J the total processing-time of the job which the new arrival finds in service, and let Y be the amount of processing-time remaining for that job at the instant of arrival. Let us first ask, what is the distribution of J?

Let $G(p)$ be the processing-time distribution function, and $E[X]$ be its expectation. Then:

$$P[p \leq J \leq p + dp] = \frac{p\,dG(p)}{E[P]}$$

or alternatively,

$$F(x) = 1/E[X] \int_0^x p\,dG(p)$$

This can be seen by realizing that the desired probability should be proportional to the long-run portion of time devoted to processing times of length p. This, in turn, should be proportional to the product of the processing-time p and the frequency $dG(p)$, with which such intervals occur. The denominator, $E[X]$ is a normalizing factor required to make the integral of this probability equal to one, since $\int_0^\infty p\,dG(p) = E[X]$.

Now to determine the distribution of Y, note that the conditional distribution of Y given that $J = p$, is the uniform distribution. That is, the observation point will fall anywhere within the processing-time interval with equal probability. So:

$$P[y \leq Y \leq y + dy | J = p] = \frac{dy}{p} \quad \text{for } 0 \leq y \leq p$$

The joint probability of landing at spot Y within an interval of length J is then:

$$P[y \leq Y \leq y + dy,\, p \leq J \leq p + dp] = \frac{dG(p)}{E[X]} \quad \text{for } 0 \leq y \leq p,\ 0 \leq p \leq \infty$$

The marginal distribution of Y is obtained by integrating over the allowable values of p. This yields the density function of Y as

$$P[y \leq Y \leq y + dy] = \int_{p=y}^{\infty} \frac{dG(p)}{E[X]} dy = \frac{1 - G(y)}{E[X]} dy \quad \text{for } 0 \leq y \leq \infty$$

To get the expectation of Y (our actual goal!) we calculate:

$$\begin{aligned} E[Y] &= \int_{y=0}^{\infty} y\, p(y)\, dy = \int_{y=0}^{\infty} y \frac{1-G(y)}{E[X]} dy \\ &= 1/E[X] \int_{y=0}^{\infty} y \int_{x=y}^{\infty} p(x)\, dx\, dy \end{aligned}$$

which is integration over a triangular area, so we can interchange the integration order as follows:

$$\begin{aligned} E[Y] &= \frac{1}{E[X]} \int_{y=0}^{\infty} y \int_{x=y}^{\infty} p(x)\, dx\, dy \\ &= \frac{1}{E[X]} \int_{x=0}^{\infty} \int_{y=0}^{x} y\, p(x)\, dy\, dx \\ &= \frac{1}{E[X]} \int_{x=0}^{\infty} p(x) \int_{y=0}^{x} y\, dy\, dx \\ &= \frac{1}{2E[X]} \int_{x=0}^{\infty} p(x)\, x^2\, dx \\ &= \frac{E[X^2]}{2E[X]} \end{aligned}$$

Example. For the Exponential distribution, the mean residual life would be

$$\bar{r} = \frac{2/\lambda^2}{2/\lambda} = \frac{1}{\lambda}$$

Does this make sense? Why?

Exercises

4-1. Prove that the Exponential Distribution is memoryless.

4-2. Given two Pareto random variables X_1 and X_2, with equal location parameters k and corresponding scale parameters α_1 and α_2,

(a) write a simple closed formula for the distribution of $\min(X_1, X_2)$.

(b) What type of distribution does $\min(X_1, X_2)$ have?

Part II

Statistical Analysis of Data

Chapter 5

Statistical Analysis of Data

“When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of science.”
— Lord Kelvin

5.1 Confidence Intervals

Definitions. CIs for the median and the mean.

5.1.1 Confidence Intervals for the Median

Based on LeBoudec.

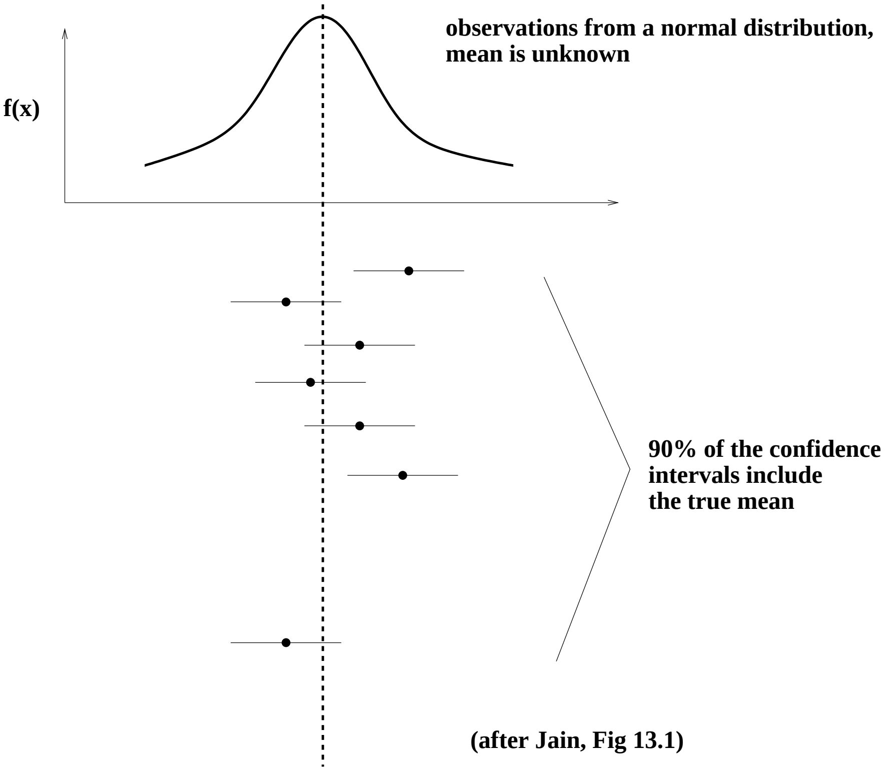
5.1.2 Confidence Intervals for the Mean

Because it is a random variable, we generally try to characterize it using mean and variance. Mean tells us what to expect, and variance tells us how much variability to expect from run to run.

Assume that there is a true value (some output parameter) that we are trying to find. This could be average value of waiting time, instantaneous queue length, etc. The metrics that we get out of the simulation are samples of a random variable, so they are drawn from some distribution. We will focus on the mean of that distribution.

We would like to find a range within which we are 90% sure that the true mean lies. What does this loose phrase "90% sure" mean? It means that we estimate that the probability that the true mean lies in the interval to be 0.9. This interval is then called the 90% confidence interval.

So based on some measurements of the metrics, we calculate a 90% confidence interval. What does this mean in practice?


observations from a normal distribution, mean is unknown
f(x)
90% of the confidence intervals include the true mean

(after Jain, Fig 13.1)

The true mean could be *anywhere* in the confidence interval range. There is no "more likely" region in the confidence interval for the mean to fall. Anotherwards, if we did this 100 times, 90 times the mean would be within the range we calculate.

Thus if two confidence intervals overlap, we cannot be sure that the means are different. The two cases therefore are indistinguishable at the confidence level we are using.

Now, how can we construct confidence intervals? We need to assess the variability of our measurements of the parameter of interest.

Can you use variance of individual measurements during a simulation run to estimate their variability? No, because the metrics are correlated over short distances as occur during a simulation run. However if we observe the simulation over long intervals, the metrics for separate long intervals will be independent. How long is long enough? Long enough for the initial state at the beginning of the interval to have little or no effect on the average metric over the interval. That is to say, long enough for a startup transient to be unimportant. The best way to achieve this is to do separate runs, each time eliminating the startup transient.

We can form an estimate of the mean of the distribution as the empirical average over all replications, and can calculate the sample variance of the respective measurements.

Now, the Central Limit Theorem tells us that the sum of a large number n of random variables, each with mean μ and variance σ^2, yields a normally distributed random variable with mean $n\mu$ and variance $n\sigma^2$. So the distribution of the average would also be normal with mean μ and variance σ^2/n. That is,

$$\bar{x} \sim N(\mu, \sigma/\sqrt{n})$$

We usually assume that the number of samples should be 30 or more for the CLT to hold. However this is a rather shaky rule of thumb – if the samples have particularly high variance, this may not be enough.

The standard deviation of the sample mean is called the standard error. The standard error is different from the population standard deviation. If the population standard deviation is σ, the standard error is only $\sigma/\sqrt{n}$. Because of the relationship on n, the variability of $\bar{x}$ decreases as we increase the number of samples n. Thus, it will turn out that using $\bar{x}$, we can get increasingly "tight" estimates of μ as we increase the number of samples n.

Now, remember that the true mean μ is a constant, while the empirical mean $\bar{x}$ is a random variable. Let us assume for a moment that we know the true μ and σ, and that we accept that $\bar{x}$ has a $N(\mu, \sigma/\sqrt{n})$ distribution. Then it is true that

$$P[\mu - k\sigma/\sqrt{n} < \bar{x} < \mu + k\sigma/\sqrt{n}] = P[-k < S < k]$$

where S is the unit normal random variable (having distribution $N(0,1)$).

We write $z_{1-\alpha/2}$ to be the $1 - \alpha/2$ quantile of the unit normal. That is,

$$P[-z_{1-\alpha/2} < S < z_{1-\alpha/2}] = 1 - \alpha.$$

So to form a 90% probability interval for S (centered on zero) we choose $k = z_{0.95}$. Turning back to $\bar{x}$, the 90% probability interval on $\bar{x}$ would be:

$$\mu - z_{0.95}\sigma/\sqrt{n} < \bar{x} < \mu + z_{0.95}\sigma/\sqrt{n}.$$

(These unit normal quantiles are tabulated in many books, for example $z_{0.95} \approx 1.65, z_{0.975} \approx 1.95,$ and $z_{0.995} \approx 3.27$.)

This is pretty straightforward so far. Now comes a tricky step. Note that the following two expressions define the same events:

$$\mu - k\sigma\sqrt{n} < \bar{x} < \mu + k\sigma/\sqrt{n}$$

and

$$\bar{x} - k\sigma\sqrt{n} < \mu < \bar{x} + k\sigma/\sqrt{n}.$$

What we are saying is, that the sample mean is in some fixed-size interval centered on the true mean, if and only if the true mean is also in a fixed-size interval (of the same size) centered on the sample mean. (Envision this geometrically.)

Note however, that in the first case, we are expressing the event that a random quantity falls between two constant bounds. In the second case we have described the same event in terms of a constant falling between two random bounds. But they are the same, and their probabilities are equal, so

$$\begin{aligned} 1 - \alpha &= P[\mu - z_{1-\alpha/2}\sigma/\sqrt{N} < \bar{x} < \mu + z_{1-\alpha/2}\sigma/\sqrt{N}] \\ &= P[\bar{x} - k\sigma\sqrt{N} < \mu < \bar{x} + k\sigma/\sqrt{N}]. \end{aligned}$$

This latter expression defines the $1 - \alpha$ *confidence interval for the mean.*

We are done, except for estimating σ. We do this directly from the data: $\hat{\sigma} = s$ (where s is the sample standard deviation, $s = \sqrt{1/(n-1)\sum(x_i - \bar{x})^2}$).

To summarize: by the argument presented here, a 100(1-α)% confidence interval for the population mean is given by

$$\bar{x} \pm z_{1-\alpha/2}\, s/\sqrt{n}.$$

Dealing With a Small Number of Samples

To do this for less than 30 samples, Let us assume that the output means are normally distributed. We have already shown that they are independent. Then we need to use Student's t distribution, which describes how a *small* number of samples from a Normal distribution tends to behave. The sample mean has a Normal distribution and the sample variance has a χ^2 distribution. The result is the t distribution.

An interval +/- ϵ within which the the true mean would fall with probability p is given by

$$\epsilon = \frac{t(n-1,p)s}{\sqrt{n}}$$

Again, $s/\sqrt{n}$ is called the *standard error*. It describes how the variance of the sample mean, as a random variable, behaves (which comes from the fact that variances sum).

In general, n = 5 or 7 are reasonable for many cases. For homework, use n = 5 and use 90% confidence intervals.

The web page uses a one-sided table, so you should use the column for 0.05 to get a 90% confidence interval.

Remember: to use Student's t distribution, you must have (approximately) normally distributed outputs. Can check this using histogram ... is it bell-shaped?

Special Topic: Dealing With Autocorrelation A complication in constructing confidence intervals is the presence of autocorrelation in system behavior. Networks and endsystems are not memoryless: they contain buffers and control algorithms that maintain past history in a way that affects current behavior. For example, when a network link is running at high utilization, routers may build up large queues which take a long time to drain. Thus the departure of a packet from the router can be delayed because of events that took place long in the past. This leads to autocorrelation in system behavior — the system's current behavior is very similar to its behavior in the recent past.

The effects of autocorrelation can be good and bad. Autocorrelation makes the near future more predictable: one can have increased confidence that if packet delays were high in recent past, they will still be high at the present time. However they also decrease the amount of information that is obtained in measurement. If one is interested in estimating the value of a system property that is strongly autocorrelated, confidence intervals will be larger than had the property been uncorrelated. Worse yet, if one constructs confidence intervals using statistical formulas that assume independence of observations, the confidence intervals will be misleadingly small leading to a false sense of precision in the results.

This can be illustrated as follows, based on the exposition in [Rou05] (which has additional valuable observations and examples). Assume we wish to estimate the mean of a process given measurements $\{X_t, t = 1, 2, ..., n\}$. The expected value of the estimator $\hat{X} = 1/n \sum_{t=1}^{n} X_t$ is the true mean $E[X]$. However, the *variance* of this estimator depends on the correlation present in the measurements.

The usual Central Limit Theorem concerns independent X_t, and states that:

$$\sqrt{n}(\hat{X} - E[X]) \to \mathcal{N}(\prime, \sigma^{\in})$$

(where $\to$ here means "converges in distribution for large T"). On this basis one can form confidence intervals for $\hat{X}$ which are proportional to $\sigma/\sqrt{n}$. However, when X_t are not independent, the estimator converges to a distribution with higher variance:

$$\sqrt{n}(\hat{X} - E[X]) \to \mathcal{N}(\prime, \int^{\in})$$

where s^2 is the *asymptotic variance.* The asymptotic variance is larger than $sigma^2$ due to the influence of autocorrelation in the X_ts. The appropriate confidence intervals are correspondingly larger, proportional to $s/\sqrt{n}$.

One can quantify this inflation in estimator variance using the autocorrelation function $r(k)$ of the underlying (continuous) process being sampled. If the process is being sampled at uniform periodic intervals δt, then:

$$s^2 = \sigma^2 \left[1 + \sum_{k=1}^{\infty} r(k\,\delta t) \right]$$

and if the process is samped at Poisson intervals with rate λ:

$$s^2 = \sigma^2 \left[\frac{1}{\lambda} \int_0^{\infty} r(u)\,du \right]$$

The autocorrelation present in network measurements can arise from a number of sources; an instuctive example is queueing delays. For an M/M/1 queue (a FIFO queue with Poisson arrivals and exponentially distributed service times), the resulting asymptotic variance is:

$$s^2 \approx \frac{4\rho^2}{(1-\rho)^4}$$

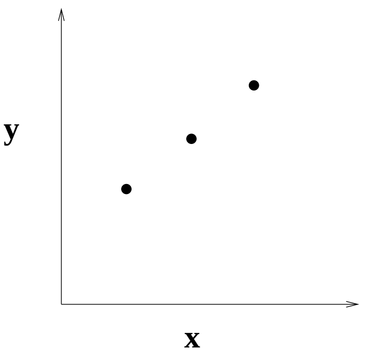
where ρ is the utilization of the queue (arrival rate times mean service time). This expression exposes an important principle: autocorrelation, and therefore asymptotic variance, increases with increasing system load. The relationship to system load is very sensitive, as shown by the fourth power in the denominator. When load is high, large queues build up, and queueing delays lead to larger autocorrelation at greater time lags k. As a result, esimates of system mean have higher variance (perhaps much higher) and are less reliable.

5.2 Interpreting Confidence Intevals

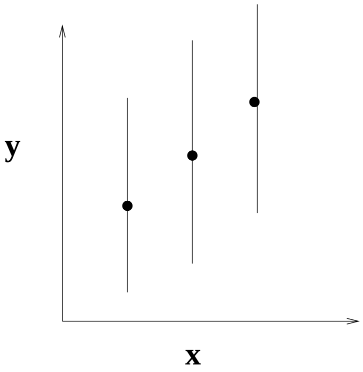
From the discussion above it should be clear that the confidence interval states that the mean could be *anywhere* within the interval. It is *not* necessarily true that the mean is more likely to be in the middle of the interval than the edges — that depends on the distribution of the estimator, which may not be symmetric. To see this, look at the figure above.

Now consider the three cases below. In each case we are plotting the output of a simulation on the *y* axis against some simulation input or parameter on the *x* axis. We wish to determine whether there is a trend in the relationship of *x* and *y*; i.e., when *x* increases does *y* also increase?

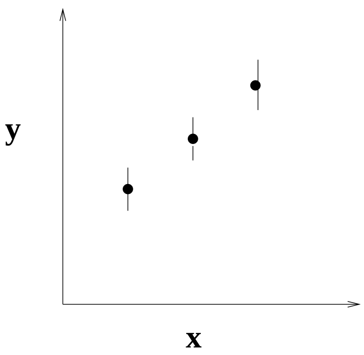
In Case 1 each point represents a single run of the simulation. In this case one *cannot* conclude



Case 1



Case 2



Case 3

that there is any relationship between *x* and *y*, because we don't know how variable the *y* values are. To conclude there is a relationship here would be the same as rolling a dice three times, observing a 1, 2, and 3, and concluding that the next roll will be 4, then 5, etc.

To address this question we add confidence intervals to our data points, resulting in Case 2. Now is it fair to conclude that there is a relationship between *x* and *y*? No! Because of the way the confidence intervals overlap, the mean could in fact be declining just as easily as increasing — remember, the mean might be *anywhere* in the confidence interval.

Finally we come to Case 3, in which we can in fact conclude there is a relationship. This is because, no matter where the mean is in each confidence interval, because the intervals don't overlap, the mean must be increasing as we increase *x*.

Part III

Palm Calculus

Chapter 6

Palm Calculus

Palm calculus is a set of tools for reconciling the time and event views of a system.

We will cover Palm calculus results for both descrete and continuous time, but we will go into detail and provide proofs only for the discrete time case.

6.1 The Inversion Formula

In working with the inversion formula, it is helpful to remember the following fact:

Let N be a random variable assuming positive integer values 1, 2, 3, ... Let X_i be a sequence of independent random variables which are also independent of N and with $E[X_i] = E[X]$ the same for all i. Then:

$$E\left[\sum_{i=1}^{N} X_i\right] = E[N]E[X]$$

6.2 Time View of a Sequence of Intervals

Also the byte view of a collection of files. Based on LeBoudec.

Introduce the notation $p_w(x)$ and $F_w(x)$ for random variable X. Define:

$$p_w(x) = \frac{x\,p(x)}{E[X]}$$

and:

$$F_w(x) = \frac{\int_0^x u\,p(u)\,du}{E[X]}$$

Perhaps these should be p_s and F_s to avoid 'waiting time' association.

6.3 Residual Lifetime

Including *Feller's Paradox*.

6.4 Mean Residual Lifetime

Exercises

Using Palm Calculus

6-1. A TCP-like Protocol. The TCP protocol, among other things, attempts to gradually increase its sending rate over time to achieve high throughput. A simplified version is the following:

- Packets are sent in *rounds*.
- The first round, one packet is sent.
- If no packets are lost in a round, then in the next round the number of packets sent is increased by 1.
- If any packets are lost in a round, then in the next round one packet is sent.

This algorithm is called *additive increase*.

After sending the packets in a round, the protocol waits for an acknowledgement from the receiver specifying whether all packets were received (not lost). The time it takes for this acknowledgement after the first packet is sent is the round-trip time (RTT). We can treat each round as lasting one RTT since the RTT is generally larger than the time it takes to send the packets.

Thus we can treat the number of packets in a round as a discrete time stochastic process $\{X_t, t = ..., -1, 0, 1, ...\}$ The events in our analysis will be the loss of one or more packets in a round.

The time instants where one or more losses occur are $\{T_n, n = ..., -1, 0, 1, ...\}$ with $T_0 \leq 0 < T_1$.

For now, we assume that all of the packets arriving in the window during which a loss takes place are considered as arriving successfully, i.e., contributing to throughput.

So some of the quantities that we will work with are $E[X_0], E[T_0], E^0[X_0], E^0[T_0], E^0[T_1]$, etc.

Show your work for each answer.

(a) Which of the quantities should be interpreted as the *throughput* of the protocol?

We'll denote the average number of rounds until a loss as n. That is, if losses occur in rounds 1,5,9 then $n = 4$.

(b) Which of the quantities should be interpreted as n?

(c) Use the inversion formula to compute the throughput in terms of n. You can assume that n is an integer here and in what follows.

Assume that packets are lost independently with probability p.

(d) Now, treat the packets in the last round that arrive before the loss as arriving successfully, while all the packets arriving after the loss as not received successfully, and so not contributing to throughput. Assume that exactly one packet in the last round is lost. Adjust the result you obtained in the last question for this correction.

(e) We make the following observation:

$$\frac{\text{losses per unit time}}{\text{packets per unit time}} = \text{losses per packet}.$$

Use this observation to derive an expression for n in terms of p.

(f) What is the throughput of this protocol expressed in terms of p?

(g) Plot the throughput of this protocol in bytes/second as a function of p, for p in the range of 0.1 to 0.001, with p on a log scale. These are typical loss rates in today's Internet. Assume an RTT of 100 ms and an average packet size of 1500 bytes (which are also typical values for the Internet). How rapidly does throughput increase with decreasing loss rate? For example, if loss rate is cut by a factor of 10, does throughput increase by the same factor?

6-2. Same problem as the last one, but we will change the protocol. Assume that instead of *adding one* to the round size on successful transmission, we *double* the round size. This is called *multiplicative increase.* Compute the resulting throughput (no need to make the last-window correction in this case), plot again as in the last question, and comment on the differences with the additive case.

6-3. Consider a network link carrying traffic for a set of users. Each user makes a request for a file, which is then sent over the link at a constant rate. Assume that the traffic demands are much less than the capacity of the link, so that every file is sent at the same rate of R bytes/second, regardless of the number of files that are simultaneously in transit.

Assume that users make requests for files according to a Poisson process with rate λ. File sizes are drawn from a distribution with density $p(x)$ having mean $\bar{x}$ bytes.

(a) What is the average time a file spends passing over the link?

Consider the time view of the link, that is, what one would see at a randomly chosen time $t = 0$.

(b) What is the average number of files in transit at time $t = 0$?

(c) State the distribution of file sizes you expect to find passing over the link at time $t = 0$.

(d) Consider the byte flowing over the link at time 0. What is the probability it comes from a file of size s or less?

6-4. Consider a Poisson process with rate λ.

(a) What is the most frequent (most likely) interarrival time?

(b) Consider an observation of the process at a random time. What is the most likely size of the interarrival time during which the observation falls?

6-5. (Based on [MSAHB04].) A particular database system implements task preemption. When a high priority task arrives, it can interrupt a low priority task. (Tasks in this system correspond to individual database queries, updates, and so forth.)

Measurements show that when a low-priority task is interrupted, the amount of work it has done is, on average, between 75% and 90% of its service demand.

(a) Explain why this is surprising. What would you have expected?

Upon further investigation, you learn more about the preemption mechanism. It turns out that tasks acquire locks as they progress, and that a task gets interrupted by a higher priority task that needs one of its locks.

(b) Assume that tasks acquire locks at a constant (uniform) rate, that all locks are held until the task completes, and that each lock is equally likely to cause a task interruption. State the expected fraction of its work that a task will have completed when it is interrupted.

(c) Compare your answer to the empirically measured value of 75% to 90%. What do you conclude about how tasks typically acquire locks? That is, are locks more likely to be acquired toward the beginning or the end of the task?

6-6. A *crossover statistic* is a rule is a rule that p fraction of the mass is contained in the $1 - p$ fraction of the largest objects. For example, "80% of the bytes are contained in the largest 20% of the files." This is sometimes called an "80/20" rule (or "90/10" etc.) It's a convenient way of expressing the tendency for a dataset to be dominated by large objects. The reason for selecting p and $1 - p$ as the cutoff values may be that it's not hard to remember: you can switch the sense and it is still true (i.e., "20% of the bytes are contained in the 80% smallest files.")

Find the crossover statistic (p) for a set of files if we assume that file sizes are drawn from:

(a) a uniform distribution on $(0, 1]$ i.e.,

$$p(x) = 1, \quad 0 < x \leq 1$$

(b) a Pareto distribution with $\alpha > 1$, i.e.,

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1} \quad x \geq k, \ \alpha > 1$$

Give a solution for $\alpha = 2$.

(c) Using the analytic results of the last part, prepare plots of the cumulative size-weighted distribution $F_w(x)$ vs the regular CDF $F(x)$ for the Pareto distribution with $k = 2$ and $\alpha = 1.5$, 1.4, 1.3, 1.2, and 1.1. That is, the plots will show the fraction y of bytes contained in the fraction x smallest files, with x and y varying from 0 to 1. Use these plots to estimate the crossover statistics for each distribution.

(d) What happens if the distribution has $\alpha \leq 1$?

6-7. What is the mean residual lifetime for a Pareto distribution with $\alpha > 2$?

6-8. You are given two Pareto random variables X_1 and X_2, with equal location parameters k and corresponding scale parameters $\alpha_1 > 1$ and $\alpha_2 > 1$. Using the results of Exercises 4-2 and 6-7, what is the mean residual lifetime for $\min(X_1, X_2)$?

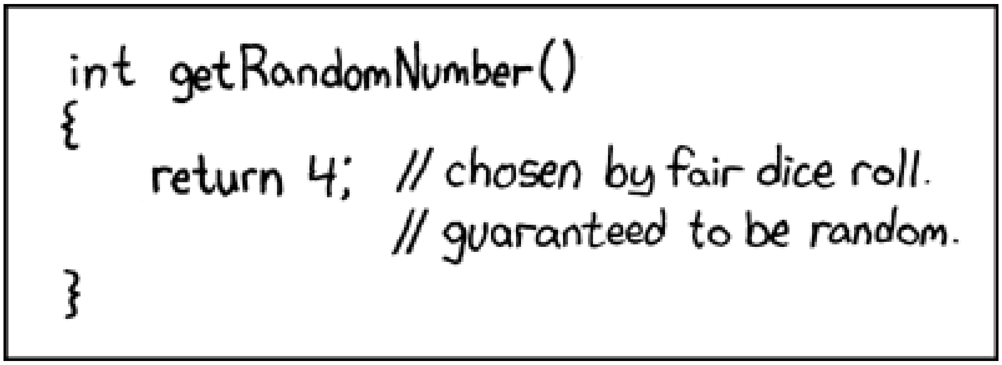
Part IV

Simulation

Chapter 7

Simulation

7.1 Generating random numbers.



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

From xkcd. `http://xkcd.com/c221.html`

Problem: no real source of randomness in a computer (because that's the way we like them!). Solution: *Pseudo*-random number generation == PRNG.

Goals of a PNRG:

1. fast,
2. long period, and
3. the "appearance" of randomness.

By the "appearance" of randomness we usually mean: the appearance of independence, and uniform distribution.

Independence is an abstract notion and can never actually be proven. Even the "appearance" of independence can only be interpreted as the ability to pass some set of statistical tests. Usually the most important test is serial correlation, so we desire our PRNG to show little correlation from one sample to the next. This is a local property.

Uniform distribution is tested for using a goodness of fit test list chi-square or K-S test. This is a global property.

> The generation of random numbers is too important to be left to chance.
> — R. Coveyou, Oak Ridge National Laboratory

7.2 Uniform PRNG

The main approach we consider is the *linear congruential method*.

$$X_{n+1} = (aX_n + b) \bmod m$$

a, b, and m are parameters of the PRNG. These are usually chosen quite carefully and it is best to use values that have been carefully studied. For our purposes we will use `drand48()` which is a linear congruential PRNG with 48 bits of internal state.

Look at the manpages for `drand48(), srand48()` and `seed48().`

X_n is the internal state of the PRNG. It completely determines what the next value output by the PRNG will be. Thus a PRNG will start repeating itself if it ever happens to return to a previously visited state.

Sometimes the output of the PRNG is not X_{n+1} but some function X_{n+1}. For example, `drand48()` outputs a 32 bit value, which is obtained from the internally stored 48 bit value X_{n+1}.

Period of a PRNG = longest nonrepeating sequence. Maximum possible period of a linear congruential PRNG is obviously m, though it could well be shorter.

Seed-setting (e.g, `srand48()`): allows control over sequence of random values used in a simulation. When debugging, it makes sense to use a particular seed for each run, to obtain identical behavior from run to run. When using simulation to get answers, need to use a different seed for each run.

Serial correlations. There is usually some slight correlation between successive values of a PRNG. A simplistic example would be the function

$$x_n = 3x_{n-1} \bmod 31$$

(this is a bad PRNG! don't use it!). If you take the output of this PRNG and plot points consisting of (x_n, x_{n+1}), you will see that they fall on three staight lines defined by: $x_n = 3x_{n-1}$, $x_n = 3x_{n-1} - 31$, and $x_n = 3x_{n-1} - 62$. Clearly we should not treat successive values as completely independent.

This is important because it shows that we need to avoid *stream-splitting.* Many times a simulation will include more that one random quantity: say, a queuing simulation in which both interarrival times and service times are random. We must not use a single PRNG stream to generate both sequences of random values, because it would introduce correlations between arrival times and service times, which would lead to erroneous results.

Another important implication is that we need to use non-overlapping streams for independent simulations as well. If we were to use overlapping streams, the simulations would repeat each other identically. So one needs to output the seed at the end of a simulation (e.g., using `seed48()`) and use it as the starting (input) seed for the next simulation run.

The most convenient way to maintain separate random number streams is using the function `erand48().` This function accepts the current state as an argument; so you can keep multiple states around, and feed whichever state you like to `erand48()` based on the particular stream you are generating.

7.3 Nonuniform RNG

7.3.1 The Inversion Method

Consider CDF $F(y)$ for RV Y. What is the distribution of $Z == F(Y)$?

$$G(z) = P[Z \leq z] = P[F(Y) \leq z]$$

Assume F^{-1} exists, then the event $F(y) \leq z$ is the same as the event $Y \leq F^{-1}(z)$, so

$$G(z) = P[Y \leq F^{-1}(z)]$$

but $P[Y \leq y] = F(y)$, so

$$G(z) = F(F^{-1}(z)) = z \quad 0 \leq z \leq 1$$

Therefore, since Y is distributed according to $F(y)$, Z will be uniformly distributed on the interval $(0, 1]$. So if we wish to generate random numbers with a given distribution $F(y)$, we need only generat uniformly distributed random numbers and apply the inverse function of $F(y)$, namely $F^{-1}(z)$, to those numbers.

Example: Generating Exponential Variates.

We wish to simulate sampling a random variable Y that is exponentially distributed, that is, has CDF $F(y) = 1 - e^{-\lambda y}$.

Invert the CDF.

$$\begin{aligned} z &= 1 - e^{-\lambda y} \\ e^{-\lambda y} &= 1 - z \\ -\lambda y &= \ln(1 - z) \\ y &= -(1/\lambda)\ \ln(1 - z) \\ y &= -(1/\lambda)\ \ln z \end{aligned}$$

So $F^{-1}(z) = -(1/\lambda)\ \ln z$. So we generate uniform random variates z on the interval [0, 1) and return $-(1/\lambda)\ \ln z$.

Not all distributions have a CDF that can be inverted.

First, if the RV is discrete, there will be flat portions of the CDF. For this case, a convention suffices: if $x1 <= urn < x2$, then return x1 as value.

Second, the RV may not have a closed form CDF (like the Poisson, Binomial, and Normal). For some such cases we can use the *composition* method.

7.3.2 The Rejection Method

Assume that pdf $f(t)$ can be majorized by some function $cg(t)$ where $g(t)$ is a pdf of a RV that we *can* generate.

For example: normal distribution, which can be (effectively) majorized by a scaled-up uniform distribution.

Then:

1. generate y with density $g(t)$ (using known method) (for example, uniform distribution)

2. generate u with uniform density on (0,1)

3. if $u \leq f(y)/(cg(y))$ then accept Y

7.3.3 Managing RNGs

In summary:

1. use a good one
2. do not subdivide a single stream ... can lead to correlations
3. use nonoverlapping streams ... manage your seeds
4. avoid using low-order bits
5. do not use unknown initialization: unknown starting point

7.4 Simulation

In discussing simulation it's important to distinguish between simulation time and run time.

The key idea behind almost all computer-based simulation is the notion of a *discrete-event* simulation.

In this context, an event is any occurrence that changes the *state* of the simulated system. This begs the question: what is the appropriate *state* of the simulated system? The state of the simulated system should clearly be smaller and simpler than the state of the true system. So the answer to this is a key modeling question, requiring judgement: what aspects of system state are relevant to the problem at hand? Remember: all models are wrong, but some are useful.

As an example, let us consider modeling a disk drive. The disk drive responds to I/O requests: reads and writes. Describing a request, one might include, among other things: the time the request is made, the location on disk, the nature of the request (read or write), the associated buffer to be emptied or filled, and in the case of the write, the particular contents of the buffer. Let us assume one is interested in the performance of the drive, but not the correctness of its operation. Then a typical modeling decision in this case would exclude the contents of the buffer is being not highly relevant: the time taken by a write operation is fairly insensitive to the actual data being written. So, in the discrete event simulation, one would not include the buffer contents as part of the state of the system being modeled.

The discrete-event appraoch suggests a particular model for organizing a simulation. Since only certain events change the state of the system, one need not simulate the system's behavior in *between* those events. So the basic operation of a discrete event simulation is to take "jumps" in time between events. This implies a key invariant: at the end of processing each event, it must be known exactly what the next event is that will affect the state of the system. It is not generally possible to process events out of time order.

This appraoch implies two key organizational components of a discrete event simulation: a “clock” to maintain the current simulated time, and an event queue to maintain a list of all of the known future events in time order. In such a scheme, the “firing” (simulated execution) of an event can cause events to be added or deleted from the event queue. (For an particular application of course, other data structures may be necessary as well.)

General structure of the control loop of a discrete event simulation:

1. Select the event with smallest time (t_e)
2. Set simulation time to t_e
3. Gather any necessary statistics
4. Fire the event, *ie*, change simulation state, possibly adding or deleting other events.

7.5 Startup Transients.

We now need to distinguish between “transient” and “steady” state. Transient: system’s *average* behavior is changing; steady state: system’s *average* behavior is staying nearly constant. Note: average behavior is a tricky notion here.

Usually the average statistics of the system will be changing rapidly early in the simulation run. (One way to think of this is that the system is unlikely to be starting in a “typical” state.) So metrics may take a while to reach “average” state.

A number of approaches are possible:

1. Start in, or close to, a “typical” state.
2. Wait until system reaches a “typical” state.

Need to plot metrics and decide when they reach steady state.

The problem is that it is not possible to define exactly what constitutes transient state vs. steady state.

Note that some variability at steady state is still likely, but decreasingly so.

So: need to do an initial run to gauge where steady state arises, and what point is OK to use as standard measurement. This should be a long run.

Then: decide based on looking at the data where to start steady state. The best simple method to do this is the truncation method:

Given a sample of n observations, the truncation method consists of ignoring the first l observations, and calcaulting the min and max of the remaining $n - l$ observations. This step is repeated for $l = 1, 2, ...n - 1$ until the $(l + 1)$th observation is neither the minimum nor the max of the remaining observations. The value of l at this point gives the length of the transient state.

7.5.1 Perfect Simulation

TBD, based on Hwk2

7.5.2 Independent Runs

Remember, you need to use nonoverlapping PRNG streams from one run to the next if you want separate runs of the simulation to be independent.

7.5.3 Interpreting Output

The output of a simulation is a random variable. Say this three times: the output of a simulation is a random variable. The output of a simulation ...

Therefore we must characterize simulation outputs; it is not sufficient or correct to simply accept the output of a simulation is “the answer” (implying determinism). That is like saying “the result when I spin a roulette wheel is the number 32.”

7.6 Pseudo-code for a simple simulation

Here is pseudocode for a simple simulation. Note that there is no “event queue” for this example because there are only two possible next events: a departure and an arrival. However in a general discrete event simulation, you need an event queue to keep track of the set of possible next events.

```
three variables:

        current_time
        next_arrival
        next_departure

/* loop until simulation is done */
while(1)
```

```
{
     if (next_arrival < next_departure)
     {
          /* next event is an arrival */

          /* skip forward in time to arrival instant */
          current_time = next_arrival;

          /* determine when the next arrival will take place */
          next_arrival = current_time + exprv(lambda);

          /* handle the arrival */
          if (a task is in service)
               put newly arrived task at end of queue
          else
          {
               place task in service
               next_departure = current_time + exprv(mu);
          }

     }
     else
     {
          /* next event is a departure */

          /* skip forward in time to departure instant */
          current_time = next_departure;

          /* handle the departure */
          if (a task is in the queue)
          {
               remove next task from front of queue
               place it in service

               /* determine when it will depart */
               next_departure = current_time + exprv(mu)
          }
          else
               /* there is no task waiting or in service */
               next_departure = infinity;
     }
}
```

Exercises

Generating Random Variates

7-1. Write a program that generates samples of a random variable drawn from an exponential distribution with parameter λ. Your program should accept λ and n as parameters, and output n values, each separated by a linefeed. You may want to check whether the output looks right by using your `mean, variance, hist,` and `edf` functions from Homework 1.

7-2. Modify your program from step 1 to generate *sums of exponentials* (also called *Erlang-k* random variables). Let E_k be the random variable that is the sum of k exponential random variables each with parameter $k\mu$. Generate 10,000 samples each from the E_1, E_2, E_5, and E_{10} distributions for $\mu = 2$. For each dataset, calculate its mean d variance, and plot its histogram. Also, plot all the edf's on a single graph.

7-3. Modify your program from step 1 to generate *hyperexponential* random variables. A two-stage hyperexponential is constructed as follows: with probability p_1, a sample is taken from an exponential distribution with parameter μ_1; otherwise (*i.e,* with probability $1 - p_1$) a sample is taken from an exponential distribution with parameter μ_2.

Generate 10,000 hyperexponential random variables for each of the following combinations of parameters:

p_1	μ_1	μ_2
0.5	1/2	1/2
0.5	1/2	1/4
0.5	1/2	1/8
0.5	1/2	1/16

For each dataset, calculate its mean and variance, and plot its histogram. Also, plot all the edf's on a single graph.

7-4. Comment on the following questions:

(a) For sums of exponentials, how do the mean, variance, and coefficient of variation change with increasing k?

(b) For hyperexponentials, how do the mean, variance, and coefficient of variation of the random variable change with decreasing λ_2?

(c) Try varying the p_1 parameter of the hyperexponential, and watch how the mean changes. What is the relationship between the mean of the two exponential distributions and the mean of the resulting hyperexponential?

7-5. Write code to generate points uniformly distributed over a unit disk. Assume that you have a routine `rand()` which generates a random number in the range $(0, 1]$.

(a) Use polar coordinates and the inversion method. That is, the program should generate two numbers: a random angle θ, $0 < \theta \leq 2\pi$, and a random radius r, $0 < r \leq 1$. Use a uniform distribution for θ, and use the inversion method to generate the r variate.

(b) Use cartesian coordinates and the rejection method. That is, the program should generate x and y such that $x^2 + y^2 \leq 1$ and the resulting points are uniformly distributed in the disk.

(c) Analyze the relative efficiency of the two methods.

Simulation

7-6. Write an discrete-event simulation of a system in which tasks arrive at a server, and are serviced in first-come-first-served order.

If a task arrives when a task is already in service, the new task is placed at the end of a service queue. When the task currently in service completes, the first task in the queue is removed and begins to recieve service immediately.

Task interarrival times are independent and identically distributed. Interarrival times are drawn from an exponential distribution with parameter λ. Thus, the arrival process is a Poisson process.

Task service times are independent and identically distributed. Service times are drawn from any of the distributions above (i.e, exponential, Erlang-k, or hyperexponential). We will use $1/\mu$ to denote the mean service time (for whichever distribution is used).

Your program should take λ and μ as input, along with random number seeds for both streams. You will want to output the random number state at the end of the program in order to use them in later runs.

Your program should collect the following statistics:

- $\bar{W_q}$: the average time spent by a task waiting in the queue
- $\bar{W}$: the average time spent by a task in the system
- $\bar{Q}$: the average number of tasks in the queue at arrival instants
- $\hat{\rho}$: the fraction of time the server was busy

Make sure that your simulation reaches steady state, and describe what you did to ensure this.

You must provide 90% confidence intervals on your results. Therefore you will need to run your simulation multiple times (at least 5) which will provide indepedent estimates of the parameters. As discussed in lecture, you will need to use Student's t distribution to calculate confidence intervals, since n is small. There is a table of the distribution on the courseinfo page.

7-7. Simulate an M/E$_k$/1 queue for varying values of k, and plot the results. Specifically, simulate the following queues:

M/E$_1$/1 (This is really an M/M/1 queue!) Set mean service time to be $1/\mu = 1/10$ and vary λ to obtain $\rho = 0.1, 0.2, 0.3, ...0.9$.

M/E$_2$/1 Set mean service time to be 1/10; this implies that service times are the sum of two exponentials each with mean 1/20. Vary ρ as before.

M/E$_5$/1 Mean service time of 1/10; vary ρ as before.

M/E$_{10}$/1 Mean service time of 1/10; vary ρ as before.

For each of the four cases plot mean time in system versus utilization (four plots), using error bars to show the confidence intervals.

7-8. Simulate an M/H$_2$/1 queue, and plot the results. In this queue, service times are drawn from a two-stage hyperexponential distribution. Simulate three cases:

p_1	$1/\mu_1$	$1/\mu_2$
1/3	1/5	1/20
3/7	1/5	1/40
7/15	1/5	1/80

Note that you have already measured the mean of hyperexponentials and determined how to compute the mean of such a random variable. Using this fact, find values of λ to obtain each ρ in 0.1, 0.2, ... 0.9. For each case, simulate the corresponding M/H$_2$/1 queue for these values of ρ. Plot average time in system as a function of system utilization for all three cases (3 plots), using error bars to show confidence intervals.

7-9. Create a summary plot, showing experimental results (all 7 curves for time in system as a function of utilization.) Label each line with the *coefficient of variation* of the service time.

7-10. Modify your M/M/1 simulation so that it *starts in steady state.* This is called ‘perfect simulation.’ For a good discussion of perfect simulation see [BV05].

The basic idea is that a typical (non-perfect) simulation starts in some fixed state, which is (1) often an unlikely state (e.g., system is empty) and (2) the same for each independent trial, which introduces bias. For example, a typical simulation starts with an empty system. This means that the initial part of each simulation run is influenced by the same (potentially unlikely) starting condition each time.

The usual response to this situation is to run the simulation for a “long” time, and to *hope* that the simulation is then in steady state, after which statistics are taken. However there is a more elegant and reliable way to address this problem, namely, to start the system in a state which corresponds to a sample at a random time of its steady-state behavior.

Most simulation parameters are defined in terms of Palm probabilities. For example, the time between arrivals, the service times, etc, are all given from the event standpoint. However to take a sample of a system at a random instant, we must know the steady-state distribution of its parameters from a time-average standpoint.

Your task is to start your M/M/1 simulation in the steady state (also called the stationary regime). To do so, answer these questions:

- What are the state variables of the M/M/1 simulation?
- What are the Palm probabilities of these random variables? (Give formulas).
- What are the time-average probabilities of these random variables? (Give formulas). Clearly explain how you got each of these answers (this is important).
- To demonstrate the effects of perfect simulation, proceed as follows. Initialize your simulation by sampling the state variables from their time-average distributions. Note that all state variables are independent so we can sample a random simulation state by sampling the state variables separately.
    - Run your original (non-perfect) simulation for 100 arrivals, and compute the four summary statistics. Do this for ρ in 0.1, 0.2, ..., 0.9, as before. Construct confidence intervals and plot the results on a single plot. Also plot the analytic predictions obtained from the classic M/M/1 formula.
    - Run your perfect simulation the same way – 100 arrivals, and compute statistics in the same manner. Plot the results and compare to analytic results as before.
- Comment on the differences you see between the two plots.
- Finally, assume you wanted to construct a perfect simulation for a M/H_2/1 queue. What challenges would you face?

Part V

System Analysis

Chapter 8

Open and Closed Systems

Recall the basis for using the Poisson process as a model of arrivals: an effectively infinite population each generating renewals at an infinitesmal rate.

When does this assumption fail? When finiteness of population is significant. That is, when the number of superimposed processes is not so large. In this case, the arrival of a customer can decrease the probability that a new customer will arrive. These are called *finite population* or *closed* queueing systems.

8.1 Setting: Networks of Queues

We work at a high level, considering systems that are comprised of *collections* of queues. We assume the system consists of M queues numbered 1 to M.

Such systems can be closed or open. If they are open, we can refer to the arrival rate λ of customers to the system, and we can relate the arrival rate at each queue $\lambda_i, i = 1, .., M$ to the overall system λ.

In contrast, a closed queueing network has no external arrivals or departures, only some fixed number of customers that circulate through the network. As a result, we usually make some queue the *reference* queue and express the arrival rate to all other queues in terms of the λ_i at the reference queue.

We will make some assumptions about how customers move between queues. Upon leaving a queue, a customer either join anothers queue immediately or leaves the system. Routing between queues is *probabilistic,* meaning that if a customer has a choice of more than one possible queue to move to next, that choice is made at random and indendently of all other routing choices.

Another assumption we will make is *job flow balance.* This means that over time, the number of jobs that arrive to the system is equal to the number that depart; furthermore, the same is true for each queue in the system. This is equivalent to assuming that the system is in steady state.

8.2 M/M/1/K/K

The simplest closed system to analyze is the M/M/1/K/K "machine repair" model. In this system customers cycle between obtaining service and waiting in a "holding area". Service is FCFS and there is a queue for customers waiting for service; service time is exponential with mean $1/\mu$. Customers waiting in the holding area are not queueed; rather, each customer independently waits some period of time O which is exponentially distributed with mean $1/\alpha$.

This is a good model for lots of closed systems: online transaction processing, timesharing a CPU among a set of processes, and of course "machine repair." In the machine repair scenario we imagine that each customer is a machine having mean time to failure of $1/\alpha$. There is a single repairperson (the server) and the time it takes to repair a machine has mean $1/\mu$.

We can model this with a finite CTMC with $K + 1$ states. The state of the CTMC captures the number of customers in the queue or obtaining service. Transitions to the left have rate μ and transitions to the right from state i have rate $(K - i)\alpha$.

This model is a birth-death model so we can immediately write down the expressions for π_i:

$$\pi_n = \frac{K!}{(K-n)!}\left(\frac{\alpha}{\mu}\right)^n \pi_0$$

and

$$\pi_0 = \left(\sum_{n=0}^{K} \frac{K!}{(K-n)!}\left(\frac{\alpha}{\mu}\right)^n\right)^{-1}$$

Connection to Fault-Tolerance. Consider the machine repair scenario. Suppose that the multiple machines are redundant. That is, the reason there are multiple machines is to increase fault-tolerance, by increasing the likelihood that at least one machine will be available. What is the probability that at any point in time there will be no machine available? Answer: π_K. For this reason $1 - \pi_K$ is called the *availability* of such a system.

Now, at this point we know everything about this system (all the π_n's) and so we could solve for the mean number of customers in the queue using the usual approach based on the π_n's. However instead of doing that, we will use a new kind of argument that will be very valuable for closed systems in general, and will shed some insight into how closed systems work. This is the *cycling customer* argument. This argument is only concerned with average values (not entire distributions, like the π_n's provide).

8.3 The Cycling Customer Argument

Consider what happens to any given customer. The customer 1) queues for service, 2) obtains service, 3) goes into "holding", and then repeats these three steps (forever). Then the average time it takes to complete one of these cycles is the sum of the averages of the three component steps:

$$\text{average cycle time} = E[O] + E[W_q] + E[X] = E[O] + E[W]$$

where we are using the usual notation that W_q is time waiting in the queue, X is service time, and W is total time waiting in the (queue + server) system.

Now, consider any point in this cycle (*e.g.,* entering the queue, entering the server, or starting to wait in the holding area). What is the *rate* at which this customer passes any such point? It is

$$\text{cycling rate of a customer} = \frac{1}{\text{average cycle time}} = \frac{1}{E[O] + E[W_q] + E[X]} = \frac{1}{E[O] + E[W]}$$

Then what is the arrival rate of customers to the queue? This is the same as the overall rate at which customers cycle through the system, which is the sum of the component rates:

$$\lambda = \frac{K}{E[O] + E[W]}$$

From this we can solve for waiting time:

$$E[W] = \frac{K}{\lambda} - E[O]$$

These last two equations are the heart of the "cycling customer" argument.

The point is, at any point in this closed loop, customers are passing at the same rate. If we can calculate the rate they pass at some particular point, that value applies everywhere.

Note that to use the cycling customer argument, we need to know either $E[W]$ or λ to obtain the other. One way to obtain λ is to go back to our CTMC solution of the system, and note that the server's utilization is

$$\rho = \lambda E[X]$$

How can we obtain ρ? Well, ρ is just $1 - \pi_0$ (as always for a single server system). So we need to know π_0 to use the cycling customer argument to obtain $E[W]$.

Note that calculating π_0 is computationally expensive: it involves summing over all K states of the system an expression that involves a factorial in K. This gets very expensive as K gets large, and in general it is true that closed systems get more computationally expensive to evaluate as the number of customers goes up.

Example ([5.2.6] from Allen). Consider a data entry shop. There are 20 operators, each with a terminal, sharing a single communication channel that is buffered. The average time to key an entry in is 80 seconds; this is approximately exponential. The average time to transmit over the shared line is 2 seconds (again, exponentially distributed). Calculate the rate at which transactions are processed, and the mean response time (time an operator spends waiting for an entry to be transmitted).

Answer. The hard part here is calculating π_0:

$$\begin{aligned}\pi_0 &= \left(\sum_{n=0}^{K} \frac{K!}{(K-n)!} \left(\frac{\alpha}{\mu}\right)^n\right)^{-1} \\ &= \left(\sum_{n=0}^{20} \frac{20!}{(20-n)!} \left(\frac{2}{80}\right)^n\right)^{-1} \\ &= 0.521\end{aligned}$$

So we can obtain $\rho = 1 - \pi_0 = 1 - 0.521 = 0.479$. Thus transaction rate is $\lambda = \rho/E[X] = 0.479/2 = 0.239$ entries per second.

Using the cycling customer argument to obtain $E[W]$:

$$E[W] = \frac{K}{\lambda} - E[O] = \frac{20}{0.239} - 80 = 3.68 \text{ secs}$$

This response time is judged to be adequate by the management, because each operator only spends $3.68/(3.68 + 80) = 4.4\%$ of their time idle waiting for order transmission.

Now, your manager tells you that they are considering adding 30 more operators to the shop, but the manager doesn't know whether this is a good idea from a productivity standpoint. Management doesn't want operators spending more than 10% of their time idle.

Let us solve the system for $K = 50$:

$$\begin{aligned}\pi_0 &= 0.0187 \\ \rho &= 0.981 \\ \lambda &= 0.490 \\ W &= 21.9 \text{ seconds}\end{aligned}$$

So with 50 operators, each operator spends 21.9/(21.9+80) = 21.5% of their time waiting for order transmission. Your boss thanks you for saving the company from this productivity pitfall and gives you the rest of the week off.

8.3.1 Extremes of the Cycling Customer Argument

Let's think about what happens in the limit cases of very many or very few customers. We can in fact analyze this without any consideration of service or think time distributions. So, this analysis applies to *all* closed single-server systems.

First, very many customers. As the number of customers goes up, the throughput (λ) must go up, but is there a limit? If there is an arbitrarily large number of customers, the server is always busy. So the rate at which they leave the server is $1/E[X]$. This tells us the rate at which customers *enter* the queue: $\lambda = 1/E[X]$.

What about response time? If we know λ, we can solve for it as before:

$$E[W] = \frac{K}{\lambda} - E[O] = \frac{K}{1/E[X]} - E[O] = K\,E[X] - E[O]$$

Note what this is saying: the waiting time is proportional to the number of customers in the system. In fact, it implies that every additional customer adds one mean service time to the wait experienced by every other customer. So, for a large number of customers, throughput is nearly constant (saturation) and response time is proportional to K.

Next, consider the case of very few customers. In this case, we expect that there is essentially no queueing taking place: any arriving customer finds the queue empty and immediately gets service. So then $E[W] = E[X]$. Solving for throughput: $\lambda = K/(E[X] + E[O])$. That is, throughput is proportional to the number of customers K. Thus, for a small number of customers, response time is nearly constant and throughput is proportional to K.

To summarize:

	Throughput (λ)	Response Time ($E[W]$)
Small No. of Customers	$K/(E[X] + E[O])$	$E[X]$
Large No. of Customers	$1/E[X]$	$K\,E[X] - E[O]$

Table 8.1: Asymptotes of the closed single-server system.

These relationships are shown graphically in Figure 8.1.

Note the special value K^*. This can be considered to be approximately where the system becomes "overloaded."

8.3.2 Example: The Apache Web Server

Figure 8.2 shows the performance of the Apache Web Server. The system studied is a 200 MHz Pentium Pro PC; the workload is generated by a closed system, with think time chosen to match

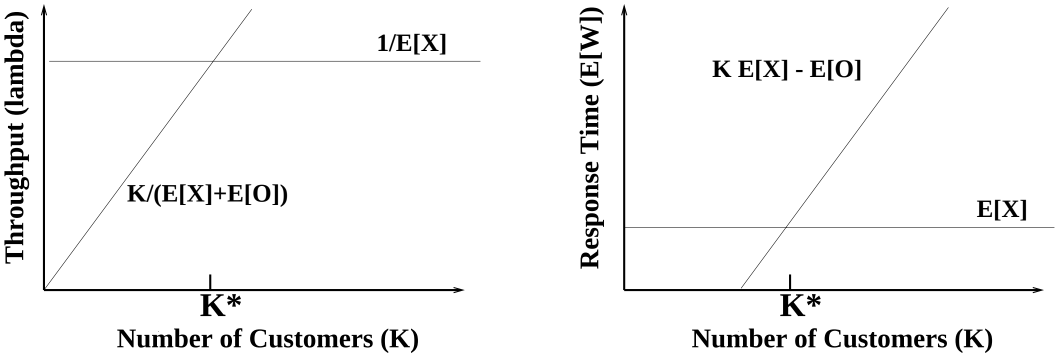


Figure 8.1: Asymptotes of the closed single-server system. Left: Throughput; Right: Response Time

empirical measurements (the workload is described in detail in [BC98]). The data in Figure 8.2 is from [BC99].

The figures show different curves for systems with varying amounts of main memory (RAM).

The throughput curves (left) show the two asymptotes: $1/E[X]$ and $K/(E[X] + E[O])$. Considering the case for 128MB (or 256MB) of RAM, we can estimate that the mean service time for a Web request in this workload mix using the many-customer asymptote as:

$$1/E[X] = 250/\text{second}$$

or

$$E[X] \approx 4\text{ms}.$$

Furthermore, we can estimate that the mean "think" time in this workload using the few-customer asymptote as

$$\text{slope} \approx 1/2 = 1/(E[X] + E[O])$$

so

$$E[O] \approx 2\text{sec}.$$

What happens as we decrease the amount of RAM in the system? The mean service time goes up, because of the decreased buffer space available for file caching, leading to increased number of disk accesses per file, which increases mean service time. This leads to smaller K^*, meaning that the server saturates at a smaller population of customers.

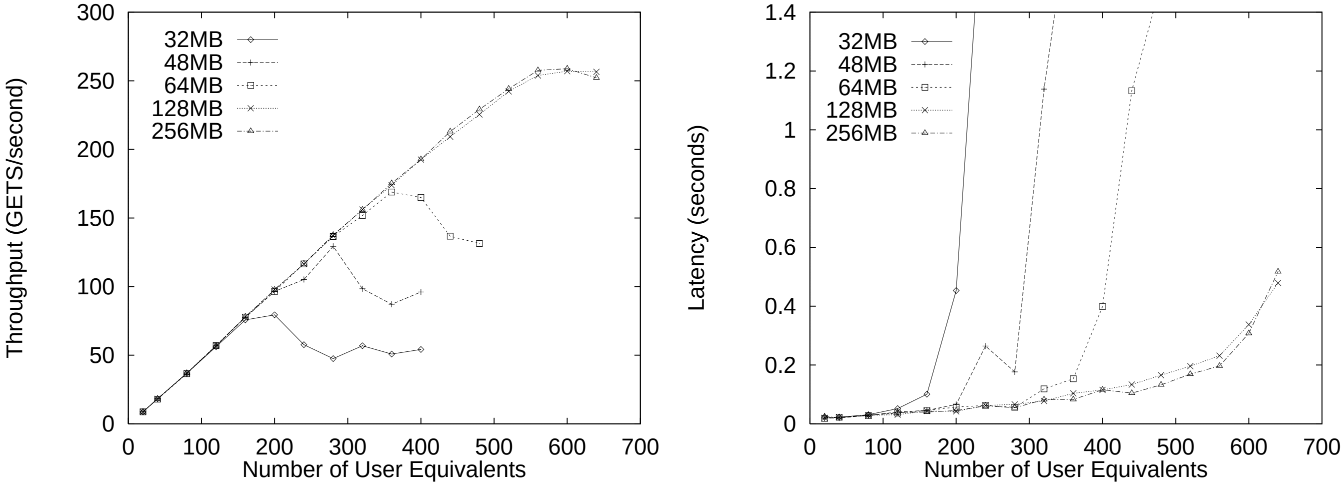


Figure 8.2: Peformance of Apache Web Server. Left: Throughput; Right: Response Time

Chapter 9

Operational Analysis

“Genius is one percent inspiration and ninety-nine percent perspiration.”
—Thomas Edison
“If Edison had a needle to find in a haystack, he would proceed at once with the diligence of the bee to examine straw after straw until he found the object of his search.... I was a sorry witness of such doings, knowing that a little theory and calculation would have saved him ninety per cent of his labor.”
—Nicola Tesla, on Thomas Edison

To get started analyzing queueing networks, we will explore a very useful method called *operational analysis*. The philosophy of operational analysis is that considerable insight into system behavior can be gained by working only with measurable quantities (as opposed to mathematical idealizations like probability distributions). This naturally leads to a set of tools that work with *average* values (and so don’t concern themselves with complete distributions of system metrics).

Let us assume that we can monitor and measure the system for some period of time T. During that time we can measure the following quantities for each queue $i, i = 1, ..., M$:

- A_i = number of arrivals to queue i during period $[0, T)$.
- B_i = amount of time server i is busy during period $[0, T)$.

From these measurable quantities we can define some average metrics:

- λ_i = A_i/T = arrival rate at queue i. Assuming job flow balance, arrival rate = departure rate = throughput.
- $\rho_i = B_i/T$ = utilization of queue i.

- $X_i = B_i/A_i$ = mean service time at queue i.

9.1 Operational Analysis Laws

Based on these definitions we can define and derive a number of useful quantities. These derivations are usually grouped into so-called “laws.”

9.1.1 The Utilization Law

$$\rho_i = \frac{B_i}{T} = \frac{A_i}{T} \times \frac{B_i}{A_i} = \lambda_i X_i$$

This is a restatement of the familiar principle; however it proceeds directly from our definitions.

9.1.2 The Forced Flow Law

Consider some number of arrivals to the reference queue (in a closed system) or from the external arrival stream (in an open system). Call this arrival set A_0.

Then it becomes useful to define the *visit ratio,* which is:

$$V_i = \frac{A_i}{A_0} \quad i = 1, ..., M$$

The visit ratio tells you, for each customer (either external arrival or arrival to queue 0), how many times it visits queue i.

Then

$$\begin{aligned} A_i &= A_0 V_i \\ \lambda &= A_0/T \\ \text{so} & \\ \lambda_i &= \frac{A_i}{T} \\ &= \frac{A_i}{A_0} \times \frac{A_0}{T} \\ &= \lambda V_i \end{aligned}$$

So we find that arrival rates throughout the system are in constant proportion, as defined by the visit ratios.

9.1.3 Bottleneck Analysis

Bottleneck analysis is a very useful way to quickly determine some basic facts about a queueing network — in particular, the maximum throughput a particular queueing network can achieve.

We start by noting that the expression for utilization

$$\rho_i = \lambda_i X_i$$

can be rewritten as

$$\rho_i = \lambda V_i X_i$$

which shows that utilization at any node is proportional to arrival rate λ. The constant of proportionality is $V_i X_i$ which represents the average total demand placed on a particular node by a single customer. That it, for each arrival to the system (or to the reference queue) a customer requires

$$D_i = V_i X_i$$

total service from node i. Then we can write

$$\rho_i = \lambda D_i.$$

This expression shows that increasing λ will increase all nodes' utilizations in proportion. As we increase λ, eventually some node will reach utilization 1. It is not possible to increase that node's utilization any more, so the system at that point is saturated.

Which node will reach saturation first? The one with highest demand D_i. That is, node j where

$$j = \arg\max_i D_i$$

is the *bottleneck* node.

What is the throughput of the system when it is saturated? The rate at which customers are serviced by the bottleneck node, *i.e.*, $1/D_j$.

9.1.4 Little's Law

Little's Law is an operational law as well. Let us define

- W_i = average waiting time of a customer at node i = average time from when a customer enters queue i to when it completes service.
- N_i = average number of customers at queue i, including the customer in service if any.

Then for each queue i,

$$N_i = \lambda_i W_i$$

9.1.5 Memoryless Service Centers

Our last law applies to memoryless service centers. Strictly speaking this is not an operational law since the memoryless property can't be directly measured (it's a property of service time distribution). However it is a simple and useful argument, and it applies to a large set of service centers.

We will define a memoryless service center as a service center with the following property: for service center i, the average waiting time of a customer W_i is $X_i(1 + N_i)$. This is true for (among other cases) FCFS queues with exponential service times, and processor-sharing queues with general service times.

For FCFS, we use a "tagged job" argument to reason as follows. Consider the average waiting time of a customer at node i. The customer must wait for the customers in the queue and the customer in service (if any). The average value of this quantity is N_i. Now, we argue that the time spent waiting for these customers that are already present upon arrival is $N_i X_i$. This is true because we will have to wait time X_i for each customer in the queue, *and* we will have to wait time X_i for the customer in service because the service center is memoryless. So the total time between arrival and departure of our tagged customer is

$$W_i = X_i(1 + N_i)$$

because the tagged customer also requires one average service time.

For PS, the argument is even simpler. After the customer arrives, there are are $(1 + N_i)$ customers in the system. So the service rate that customer gets is $1/(1 + N_i)$ and the time it takes until the customer completes service is

$$W_i = X_i(1 + N_i).$$

Now let us use this expression along with Little's Law to solve for N_i.

$$\begin{aligned} N_i &= \lambda_i W_i \quad \text{(Little's Law)} \\ &= \lambda_i X_i(1 + N_i) \quad \text{Substituting expression for } W_i \\ &= \rho_i(1 + N_i) \\ &= \frac{\rho_i}{1 - \rho} \end{aligned}$$

So, concerning ourselves only with mean values, $N_i = \rho_i/(1 - \rho_i)$ whenever service centers are memoryless. Note that this is the same expression as for the M/M/1 queue; however our assumption here is much more general: only a memoryless service center.

9.2 Bottleneck Analysis of Closed Systems

Looking at the output of the MVA algorithm (next section) λ or W as a function of number of customers N it is clear that for large N and small N there are certain asymptotes. In fact, these asymptotes apply to *any* closed system, and so are quite useful.

The general style of this analysis is similar to the M/M/1/K/K analysis in which we used the "cycling customer" argument.

We can formulate this argument as an operational law. Divide the system into two parts: the part in which there is load-dependent delay (like queueing) and the part in which there is only load-independent delay (like "think time"). For example, in the BCMP system the load-dependent service centers would the FCFS, LCFS, and PS centers; and the load-independent part would be the IS centers.

The mean delay a user encounters is the load-dependent part of the system is called W ("response time") and in the load-independent part of the system is called Z.

The operational law is called the "interactive response time" law. We start by observing that a single user cycles once through the system in time $1/(W + Z)$. Then the N users are cycling at rate $N/(W + Z)$.

9.3 Analyzing Closed Systems with MVA

The previous techniques for calculating average values of response time apply to open systems in which the arrival rate of custoemrs is known. However, for closed systems, calculating response time is more difficult, because circulation rate of customers is not necessarily known (only number of customers in system is known).

The difficulty in deriving response time in a closed system is due to the fact that response time is a function of circulation rate, and vice versa. We can analyze the limiting cases of very many customers, and very few customers, but for intermediate cases the problem is difficult.

There are methods for complete solution of closed systems (*i.e.*,, calculating the joint probability distribution of the states of each of the queues in the system) but they are beyond our scope. One issue is that in a system with K customers and N queues the state space is of size C^{N+K-1}_{N-1} which grows exponentially in K, and so often precludes computational solution.

Fortunately, there is a simply way to build up to the intermediate cases, if we restrict ourselves to working only with average metrics. The method is called *mean value analysis* or MVA.

MVA is concerned with networks of queues that are *memoryless.* There are four queueing systems that are memoryless:

1. exponential service times and FCFS service;
2. exponential service times and LCFS service;
3. general (any) service times and PS service;
4. infinite-server models (pure delay systems) with exponential service times.

This is by no means all the possible queues one might encounter in practice (in particular M/G/1/FCFS is out) but it is a useful set nonetheless. The most important case is PS, because PS is a good model for lots of real world systems and in the case of PS, service time distribution doesn't matter.

When a queueing network is composed of memoryless service centers and uses probabilistic routing, it is called a *product form network.*

There are two key ideas in MVA:

1. Little's Law can be applied to each queue in the network, but also to the network as a whole
2. The Mean Value Theorem (due to Lavenberg and Reiser), which states that a customer arriving to a queue in a product form network sees the same distribution of customers as an outside oberserver would see if one less customer were circulating in the network.

The idea then of MVA is: to compute the behavior of a system with K customers, start from the case of an empty system and add customers one by one. For K customers:

1. For each queue, note the average queue length if one less customer were present (*i.e.*, use the results from step $K - 1$). Turn this into average waiting time using mean service time.
2. Total waiting time for a customer is then weighted average of waiting times in each queue. That is, calculate the average throughput (cycling rate) for the system by applying Little's Law to the entire network; and
3. Calculate the average number of customers in each queue by applying Little's Law to the average throughput and average waiting time from steps 1 and 2.

Here is the MVA algorithm more precisely.

The Basic MVA Algorithm

Inputs:

- n (the number of customers)
- m (the number of servers)
- for each server:
    - $\bar{x}_i$ (mean service time for queue i)
    - V_i (visit ratio for queue i)
    - t_i (type of service discipline at queue i: FCFS, PS, IS or LCFS)

Outputs:

- λ (throughput when n customers)
- R (response time when n customers)

Variables:

- $l_i(n)$ (the average number of customers at queue i when there are n customers in the system)
- w_i (the mean time spent waiting in the queue at node i)

Algorithm:

1. Let $l_i(0) = 0$ for all i
2. For $j = 1$ to n:
    (a) For all i:

$$w_i = \begin{cases} 0 & \text{for IS} \\ \bar{x}_i l_i(j-1) & \text{for FCFS, PS, LCFS} \end{cases}$$

    (b)

$$R = \sum_{i=1}^{m} (w_i + \bar{x}_i) V_i$$

(c)

$$\lambda = \frac{j}{R}$$

(d) For all i:

$$l_i(j) = V_i \lambda (w_i + \bar{x}_i)$$

Exercises

Operational Analysis

9-1. Consider a queueing network with the following characteristics. The network consists of a CPU and two disks (Disk A and Disk B). We observe the system for one hour (3600 seconds) and make the following measurements:

Jobs passing through the system:	10,800
CPU busy time:	1728 seconds
Disk A busy time:	1512 seconds
Disk B busy time:	2592 seconds
I/Os at A:	75,600
I/Os at B:	86,400

Jobs route through the system such that each entering job first visits the CPU; then it visits one of the two disks. After each visit to a disk, the job returns to the CPU. Jobs leave the system after one of their visits to the CPU.

Questions to answer about this system:

(a) Basics. What is the throughput λ; visit ratios V_A, V_B, and V_{CPU}; per-job demand at each node D_A, D_B, and D_{CPU}; utilization of each node ρ_A, ρ_B, and ρ_{CPU}; and mean service time at each node X_A, X_B, and X_{CPU}?

Assume the CPU scheduling is processor-sharing, and that service at the disks is FCFS with exponential service times.

(b) Response time. What are the per-node response times W_A, W_B, and W_{CPU} and the system response time W?

We will call this system the *base case.* Now consider three possible modifications of the base case:

(c) Assume load goes up by 1/3, *i.e.*, $\lambda_{new} = 4/3\lambda$. What is the new response time?

(d) What if Disk B were to fail so that all requests go to Disk A? What is the new response time?

(e) Let us say we add a cache to speed up Disk B. The cache hit rate is 50%. Using the cache increases CPU time by 30% and increases I/O time on Disk B by 10%. V_{CPU} does not change, nor does V_A; however V_B goes down by 50%. What is the new response time?

Part VI

Markov Chains

Chapter 10

Discrete Time Markov Chains

Now interested in mathematical models of systems that can capture essential behavior. We will focus on discrete-valued random processes, i.e., chains. Recall definitions in Section 4.1.

The values that the random variables in the chain can take on will be (as usual) denoted in lower case (*e.g.,* x, x_0, etc.). These values will be referred to as *states.* Therefore a realization of a discrete state space process is a sequence of states. We think of a chain as passing through a sequence of states. Passing from one state to the next is called a *transition.*

What kind of structure can a stochastic process have? A stochastic process is distinguished from an arbitrary set of RV's by its ordering, so we should think about the effect of order on the RVs. The simplest structure is i.i.d. RVs, in which order is irrelevant (why?). However in most cases, this model is insufficient for capturing essential behavior, because a real system's evolution forward in time depends on its current state.

Consider the most general form of stochastic process: complete dependence on past, *e.g.,* for a discrete-valued process:

$$P[X_{n+1} = x_k | X_n = x_i, X_{x-1} = x_j, ..., X_1 = x_l]$$

In this case, to determine the probability of a particular state x_k being the next state, we would need to know all of the previous states. This case, while very general, is very hard to work with analytically.

A common compromise is to consider so-called *Markov* processes. Consider the sequence of states x_i that the system passes through.

Definition. If the conditional probability

$$P[X_{n+1} = x_k | X_n = x_i, X_{n-1} = x_j, ..., X_1 = x_l] = P[X_{n+1} = x_k | X_n = x_i],$$

for any $x_i, ...x_l$ then $\{X_n\}$ is a *Markov Chain* (Markov 1856-1922).
The chain is said to have the *Markov property.*

The Markov property refers to the fact that the next state depends *only* on the current state. The Markov property means that the future behavior of the system is independent of the past, and affected only by current state.

Furthermore, if $P[X_{n+1} = j | X_n = i] = P_{ij}$ does not depend on time n, it is called a *homogeneous* Markov chain (or a MC w/ constant transition probabilities). We will restrict ourselves to this case and so P_{ij} will always be defined. This means that the system "behaves the same" regardless of what time it is.

Question. What is the difference between homogeneity and stationarity? Is every homogeneous Markov Chain stationary? Is every stationary Markov Chain homogeneous?

The model is general enough to handle many interesting things. Why? Because for many computer sytems, the Markov property is a good approximation to system behavior. Consider a communication network that is carrying traffic for some number of users K_n at time n. It's a good assumption that knowledge of K_n gives some information about the likelihood of possible values of K_{n+1}. However, knowledge of *both* K_n and K_{n-1} probably doesn't give much more information about K_{n+1} than does just knowledge of K_n.

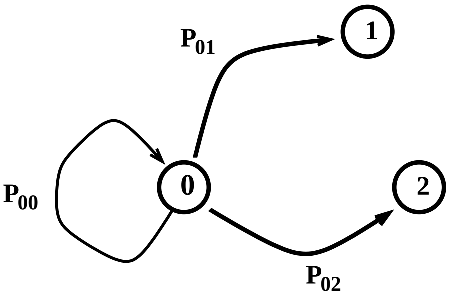
Aside. Informally, such systems are considered to have *no memory* (or "very little" memory). Recall our (earlier) discussion of autocorrelation and memory. Measurements of a system with little memory will show low levels of autocorrelation. This makes clear when the Markov property is an inappropriate model: when a system shows long memory. For example, consider the instantaneous utilization of the communication network at time n, say R_n. Because utilization of a communication network often shows long memory (as explained earlier) the Markovian assumption is probably a bad one for R_n.

Note that a Markov Chain is defined in terms of the event view. That is, the Markov Chain is defined in terms of conditional probabilities, where the condition is the arrival in a particular state (an event). Once again, we will be defining a system in terms of the event view, while the fundamental questions we will ask will concern the random or time-average view.

Discrete Time

For discrete time, each transition is a step; transitions can occur back to the same state.

Some examples in graphical representation: vertices and arcs.



We model evolution of the chain as a series of steps. At each step, *some* transition must occur therefore $\sum_j P_{ij} = 1$.

Stochastic Matrices

Another representation of a discrete time MC is as a matrix, called the *transition probability matrix* and denoted $\hat{P}$.

$$\hat{P} = \begin{bmatrix} P_{00} & P_{01} & P_{02} & \dots & P_{0j} & \dots \\ P_{10} & P_{11} & P_{12} & \dots & P_{1j} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{i0} & P_{i1} & P_{i2} & \dots & P_{ij} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

This is not an arbitrary matrix:

1. All entries are non-negative.
2. All rows sum to 1. A square matrix in which rows sum to 1 is called a *stochastic matrix.*

Question. Do the columns sum to 1? Why or why not?

Note that since all rows sum to 1, so if you were given $n-1$ columns you could compute the missing column.

Definition. n*-step* transition probabilities P^n_{ij} are defined as follows:

$$P^n_{ij} = P[X_{n+m} = j | X_m = i] \quad n \geq 0, \;\; i, j \geq 0$$

Of course $P^1_{ij} = P_{ij}$.

Example Markov Chains

Random walk. Set of states is all integers. Probability of moving to the right is p, to the left is $q = 1 - p$.

$$P_{i,i+1} = p; \;\; P_{i,i-1} = q \;\; (i = 0, \pm 1, \pm 2, ...)$$

Two representations: as set of states, or as a matrix:

$$\hat{P} = \begin{bmatrix} \ddots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & q & 0 & p & 0 & 0 & \dots \\ \dots & 0 & q & 0 & p & 0 & \dots \\ \dots & 0 & 0 & q & 0 & p & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

This matrix is infinite with a particular structure: two nonzero diagonals.

Question. Is this Markov Chain stationary? Under what conditions?

Random walk with absorbing barriers. Similar case as before, but the chain is finite. Notes N and $-N$ have self-loops with probability 1. So:

$$P_{i,i+1} = p; \;\; P_{i,i-1} = q \;\; (i = 0, \pm 1, \pm 2, ..., \pm N - 1)$$
$$P_{NN} = P_{-N-N} = 1$$

$$\hat{P} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ q & 0 & p & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & q & 0 & p \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 \end{bmatrix}$$

$-N, N$ are called absorbing states.

Question. Is this Markov Chain stationary? Under what conditions?

Definition. If $P_{ii} = 1$, state i is an absorbing state.

More complicated example.

$$\hat{P} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{2} \end{bmatrix}$$

Try drawing this. How does the structure of the matrix relate to the structure of the chain as a graph?

10.1 Chapman-Kolmogorov Equations

We can calculate the n-step transition probabilities using the *Chapman-Kolmogorov equations:*

$$P_{ij}^{n+m} = \sum_{k=0}^{\infty} P_{ik}^{n} P_{kj}^{m}$$

This can be understood by noting that $P_{ik}^{n} P_{kj}^{m}$ representse probability that starting in state i the process will go to state j in $n + m$ transitions through a path which takes it into state k at the nth transition. Hence, summing over all intermediate states k yields the probability that the process will be in state j after $n + m$ transitions.

If we let $P^{(n)}$ denote the matrix of n-step transition probabilities, P_{ij}^{n}, then these equations tell us that

$$P^{(n+m)} = P^{(n)} \cdot P^{(m)}$$

where the dot represents matrix multiplication.

Example. (Ross) Suppose the chance of rain tomorrow depends on previous weather conditions only through whether or not it is raining today; *i.e.*, past weather conditions do not influence tomorrow's weather. Suppose then that if it rains today, then it will rain tomorrow with probability 0.7; and if it does not rain today, then it will rain tomorrow with probability 0.4.
We will let state 0 represent rain and state 1 represent no rain. Then the associated Markov Chain looks like:

$$\hat{P} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

Now, calculate the probability that it will rain four days from today, given that it is raining today. By definition,

$$P^{(1)} = \hat{P} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

So,

$$P^{(2)} = P^{(1)} \cdot P^{(1)} = \begin{bmatrix} 0.61 & 0.39 \\ 0.52 & 0.48 \end{bmatrix}$$

And,

$$P^{(4)} = P^{(2)} \cdot P^{(2)} = \begin{bmatrix} 0.5749 & 0.4257 \\ 0.5668 & 0.4332 \end{bmatrix}$$

So the desired probability $P_{00}^{(4)}$ equals 0.5749.

Numerical Example:

```
>> P = [[0.7 0.3];[0.4 0.6]]

P =

    0.7000    0.3000
    0.4000    0.6000

>> P^2

ans =

    0.6100    0.3900
    0.5200    0.4800

>> P^4

ans =
    0.5749    0.4251
    0.5668    0.4332

>> P^8

ans =

    0.5715    0.4285
    0.5714    0.4286

>> P^16

ans =

    0.5714    0.4286
    0.5714    0.4286
```

Does this work for any matrix? Let's try a random stochastic matrix. (Note the steps below; why do we need the second and third?)

```
>> R = randn(5,5)
>> R = abs(R)
>> for i = 1:5
R(i,:) = R(i,:) / sum(R(i,:))
end

>> R
>> R^2
>> R^16
```

Unconditioning using Initial State

So far we have been considering only conditional probabilities, *e.g.*, $P_{ij}^{(n)}$ is the conditional probability of being in state j at time n *given* that the initial state at time 0 is i. We can find the unconditional probability of being in the state at time n, if we know the initial probability of being in each state at time 0. Let us say the initial probability of being in state i at time 0 is α_i.

The set of states forms a partition of all possible cases at time 0. So by the law of total probability,

$$\begin{aligned} P[X_n = j] &= \sum_i P[X_n = j | X_0 = i] \, P[X_0 = i] \\ &= \sum_i P_{ij}^{(n)} \alpha_i \end{aligned}$$

For instance, in our previous example, if $\alpha_0 = 0.4$ and $\alpha_1 = 0.6$, then the (unconditional) probability that it will rain four days after today is

$$\begin{aligned} P[X_4 = 0] &= 0.4 \, P_{00}^{(4)} + 0.6 \, P_{10}^{(4)} \\ &= (0.4)(0.5749) + (0.6)(0.5668) \\ &= 0.5700 \end{aligned}$$

Looking at these examples, the question arises, what is the nature of $P_{ij}^{(n)}$ as n goes to infinity? Does it approach some value? Does that value depend on i? To answer these questions we must lay more groundwork.

State Classification

Communication; Classes; Irreducibility.

If $P_{ij}^n > 0$ for some $n > 0$, then j is *accessible* from i. This implies that state j is accessible from state i if and only if, starting in i, it is possible that the process will ever enter state j.

If the reverse is true as well, then i and j *communicate*.

Communication is a transitive property, so all communicating states form a class. So the concept of communication divides the state space up into a number of separate classes. If whole chain forms one class, then it is called *irreducible.*

Recurrence.

For any state i we let f_i denote the probability that starting in state i, the process will ever re-enter state i.

State i is *recurrent* if $f_i = 1$ and *transient* if $f_i < 1$. Note that if state i is recurrent then the process will re-enter state i an infinite number of times.

On the other hand, suppose state i is transient. Each time the process enters state i, there will be a positive probability, namely $1 - f_i$, that it will never again enter that state. Therefore, starting in state i, the probability that a process will be in state i for exactly n future time periods has a geometric distribution with parameter f_i. So if a state is transient then the expected number of visits to that state is finite.

Finite chains must have some recurrent state. On the other hand, an infinite chain can have all states transient.

If one state in a class is recurrent, then all must be recurrent (can you prove this?). So recurrence/transience is a class property: all states in a class either have it or don't.

Periodicity.

State i is said to have *period* d if $P_{ii}^n = 0$ whenever n is not divisible by d, and d is the largest integer with this property.

For example, starting in i, it may be possible for the process to enter state i only at times 2, 4, 6, 8, ... in which case state i has period 2.

A state with period 1 is called *aperiodic*. It can be shown that periodicity is also a class property. That is if state i has period d, and states i and j communicate, then state j also has period d.

Positive Recurrence.

If state i is recurrent, then is it is said to be *positive recurrent* if, starting in i, the expected time until the process returns to state i is finite. It can be shown that positive recurrence is a class property.

Positive recurrence of state i means that, not only do the probabilities of returning to state i sum to 1 (recurrence):

$$\sum_{n>0} P[\text{going from } i \text{ to } i \text{ in } n \text{ steps, but no less}] = 1,$$

but also that the expected number of steps until the chain returns is finite:

$$\sum_{n>0} nP[\text{going from } i \text{ to } i \text{ in } n \text{ steps, but no less}] < \infty.$$

Informally, if we watch the chain for a fixed duration, we expect it to return to state i some number of times that is proportional to the duration of observation.

It can be shown that in a finite-state Markov chain all recurrent states are positive recurrent. Only an infinite state Markov chain can lack this property, and then only if the probability of returning in exactly n steps (and not before) declines very slowly in n — which will not be the case for any of the chains we consider.

Finally, we can state the key property: positive recurrent, aperiodic states are called *ergodic*. Ergodicity is therefore a class property. If the Markov Chain is irreducible and ergodic, then the chain itself is ergodic. As we have seen, this means that ensemble properties (properties of a representative random variable) are the same as properties of a single realization over time. So, to study time averages of such a chain we can study the averages of representative random variables.

The Bottom Line. In summary, we are concerned with ergodic Markov chains. These are those in which one will always eventually go from any given state to any other state (in finite expected time), and which don't show any periodic (cyclical) behavior.

10.2 Limiting Probabilities

Now let's return to the question of in what cases will the rows of a Markov Chain become similar for a large number of transitions, as seemed to be happening in the rainy / not rainy example.

Theorem. (stated without proof.) For an ergodic Markov chain, $\lim_{n\to\infty} P_{ij}^{(n)}$ exists and is independent of i. Furthermore, letting

$$\pi_j = \lim_{n\to\infty} P_{ij}^{(n)}, \quad j \geq 0$$

then π_j is the unique nonnegative solution of

$$\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}, \quad j \geq 0$$

$$\sum_{j=0}^{\infty} \pi_j = 1.$$

This can be expressed as

$$\pi^T P = \pi^T,$$

$$\pi^T \mathbf{1} = 1.$$

Linear Algebraic Refresher: The equation $\pi^T P = \pi^T$ is called the *eigenproblem.* This can be also be written as $P^T \pi = \pi$. The vector π is called an *eigenvector* of the matrix P^T. Associated with each eigenvector e_i of P^T is a corresponding eigenvalue λ_i such that $P^T e_i = \lambda_i e_i$.

Note that these equations do not uniquely determine the values of the steady-state probability vector π. However if we add the constraint that $\sum \pi = 1$ then the set of equations has a unique solution.

Aside. The eigenproblem-view is not the only way to look at this. One can also think of it as a set of simultaneous linear equations:

$$\pi^T(I - P) = 0$$

This view of the Markov Chain at steady state emphasizes that the *net probability change* from any state to the set of all other states is zero. We will return to this idea in more detail later.

The Numerical View

All of this can be cast in a numerical framework based on $P^T\pi = \pi$. We can construct any finite m-state DTMC and express it as an $m \times m$ matrix P. This can then be solved using a numerical package (*e.g.,* matlab, mathematica, octave, Splus, R, ...).

In solving the eigenproblem for the matrix P^T, one obtains a full set of $m - 1$ eigenvectors. The vector π is the *principal* eigenvector, that is the eigenvector corresponding to the largest eigenvalue. Clearly π needs to be an eigenvector, but why this particular one? The answer comes from the action of P as a linear operator.

Let's take some random starting point $x_1 \in R^m$. Since P is of rank m it has m distinct eigenvectors $e_i \in R^m$. Let's express x_1 in terms of those eigenvectors: $x_1 = \sum_{i \in 1..m} \alpha_i e_i$ Now take "one step forward" in the Markov chain, ie, set $x_2 = x_1 P$. Now by the action of P, x_2 will be $x_2 = \sum \lambda_i \alpha_i e_i$ where λ_i is the eigenvalue corresponding to eigenvector e_i. Repeating this process will yield $x_n = x_1 P^{(n)} = \sum \lambda_i^n \alpha_i e_i$. In this sum, obviously the term with the largest λ_i will dominate as n grows large. (This is the so-called *Power* method for constructing the principal eigenvector).

So the limiting distribution π is the principal eigenvector of P. To obtain this in, *e.g.,* matlab or octave:

```
P = [[0.7 0.3];[0.4 0.6]]
PT = P'
[V, Lambda] = eig(PT)
pi = V(:,1)/sum(V(:,1))
```

This P matrix corresponds to the rainy / not-rainy example.

One has to transpose P first because these packages define the eigenproblem as $Me = \lambda e$, ie, they assume vectors are column vectors. Likewise the columns of V are the resulting eigenvectors. The last step normalizes the eigenvector to sum to 1 (*i,e.,* enforces the law of total probability).

Note that any square matrix with rank m will have m nonzero eigenvalues. However, it is only matrices corresponding to *irreducible ergodic* Markov chains that will have a *single largest eigenvalue with value 1.*

Example: TCP's Window Size

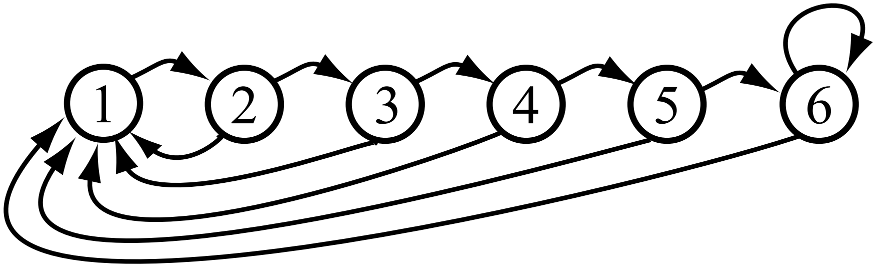
As a first numerical example, let's consider the following problem. TCP is the transport protocol used in the Internet; all data flowing reliably in the Internet (*e.g.,* Web pages) is sent via TCP. One of TCP's jobs is to control the rate of data flow, so that if congestion arises, TCP reduces its sending rate, until the traffic jam clears. On the other hand, TCP tries to send as fast as it can when there is no congestion. TCP's sending rate is controlled by its *window size* which is the number of packets that are "in flight" between the sender and receiver at any time.

A (very) simplified view of TCP's control algorithm to do this is as follows:

```
1. Start with window size = 1
2. Send a window's size number of packets
3. Did they all arrive at the recipient?
   (a) If so, have we reached the maximum window size?
        i. If so, keep the window size the same, and go to Step 2.
       ii. If not, increase the window size by 1, and go to Step 2.
   (b) If not, go to Step 1.
```

Let's assume that the probability that an entire window's worth of packets does *not* arrive at the recipient is p. So p is the probability that at least one packet in a window's worth is lost. (Question: is this a good model?) Furthermore, let's assume that these loss events are independent. Then TCP's control algorithm can be modeled as a Markov Chain.

The states of the chain correspond to window sizes. Let's assume that the maximum window size is 6. Here is a picture of the DTMC. Arcs to the right have probability $1 - p$ and arcs to the left have probability p.



The first question we'll ask is: what is the steady-state distribution of window sizes?

Here is some Matlab/Octave code to answer that question, for $p = 0.01$.

```
makep = @(p) [[p 1-p 0 0 0 0];
   [p 0 1-p 0 0 0];
   [p 0 0 1-p 0 0];
   [p 0 0 0 1-p 0];
   [p 0 0 0 0 1-p];
   [p 0 0 0 0 1-p]]

P = makep(0.1)

[V, lambda] = eig(P')

% Note that in this case, lambda==1 shows up in column 6
pi = V(:,6)/sum(V(:,6))

bar(pi)
```

Now let's ask this: what is the average window size?

Answer: $E[\text{window size}] = \sum_{i=1}^{6} i \cdot \pi_i$

Which in matlab is just:

```
[1 2 3 4 5 6] * pi
```

Now think about this: assume that each window's worth takes the same amount of time to send (this is not a bad approximation). Let's call this time R. Typical values of R are from 0.1 sec to 0.5 sec. Then the rate that TCP is sending is equal to the window size divided by R. What is the average rate in packets per second that TCP sends when $p = 0.01$ and when $p = 0.10$?

Example: Google's Page Rank Algorithm

From [BP98]:

> *We assume page A has pages $T_1 ... T_n$ which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d*

> *to 0.85. There are more details about d in the next section. Also $C(A)$ is defined as the number of links going out of page A. The PageRank of a page A is given as follows:*
>
> $$PR(A) = (1 - d) + d(PR(T_1)/C(T_1) + ... + PR(T_n)/C(T_n))$$
>
> *Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.*
>
> *PageRank or $PR(A)$ can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web. Also, a PageRank for 26 million web pages can be computed in a few hours on a medium size workstation. There are many other details which are beyond the scope of this paper.*
>
> *[...]*
>
> *PageRank can be thought of as a model of user behavior. We assume there is a "random surfer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank. And, the d damping factor is the probability at each page the "random surfer" will get bored and request another random page. One important variation is to only add the damping factor d to a single page, or a group of pages. This allows for personalization and can make it nearly impossible to deliberately mislead the system in order to get a higher ranking. We have several other extensions to PageRank, again see [Page 98].*
>
> *Another intuitive justification is that a page can have a high PageRank if there are many pages that point to it, or if there are some pages that point to it and have a high PageRank. Intuitively, pages that are well cited from many places around the web are worth looking at. Also, pages that have perhaps only one citation from something like the Yahoo! homepage are also generally worth looking at. If a page was not high quality, or was a broken link, it is quite likely that Yahoo's homepage would not link to it. PageRank handles both these cases and everything in between by recursively propagating weights through the link structure of the web.*

To interpret this in terms of the language we have been using in this course, please read the first five pages of [LM05], available from the courseinfo page.

Example from Deeper Inside PageRank [LM05]:

```
Step 1.

>> P = [[0 1/2 1/2 0 0 0];
[0 0 0 0 0 0];
[1/3 1/3 0 0 1/3 0];
[0 0 0 0 1/2 1/2];
[0 0 0 1/2 0 1/2];
[0 0 0 1 0 0]]

Step 2.

P(2,:) = [1/6 1/6 1/6 1/6 1/6 1/6];

or

P(2,:) = ones(1,6)/6;

Step 3.

E = ones(6,1)
M = E * E’
M = M / 6

alpha = 0.90

Pbar = alpha * P + (1 - alpha) * M

What is steady state?

[V, lambda] = eig(Pbar’)

% Note that in this case, lambda==1 shows up in column 6
pi = V(:,6)/sum(V(:,6))

>> pi= V(:,1)/sum(V(:,1))

pi =

    0.0372
    0.0540
    0.0415
    0.3751
    0.2060
    0.2862
```

10.3 The Reversed Chain

Imagine that you observe a particular realization of an ergodic DTMC. For example, let us say that the DTMC has states denoted by integers $1 \leq n \leq 10$. After the chain has been running for a long time (so that its current state is idependent of the starting state, ie, it is in the stationary regime) you observe a long sequence, eg,:

$$..., 3, 9, 8, 8, 1, 2, 1, 7, 3, 4, 3, 5, ...$$

Clearly, one can infer P_{ij} by counting the relative frequency of transitions from i to j (this is the case due to the ergodicity of the chain).

Now imagine that you looked at the same sequence in reverse order. Does it always correspond to a realization of some Markov Chain?

More formally, given a (forward) Markov Chain

$$..., X_n, X_{n+1}, X_{n+2}, X_{n+3}, ...$$

is the reversed stochastic process

$$..., X_{n+3}, X_{n+2}, X_{n+1}, X_n, ...$$

also a Markov Chain?

The answer is yes, and the reason is intuitive: since the current state of the Markov Chain is independent of the past, the future state of the reversed chain must be independent of the current state. For the reversed chain, this is a Markov property.

Again, more formally, let's establish the Markov property for the reversed chain:

$$P[X_n = j | X_{n+1} = i, X_{n+2} = k, ...] = P[X_n = j | X_{n+1} = i]$$

Let's just consider one step in the past:

$$\begin{aligned} P[X_n = j | X_{n+1} = i, X_{n+2} = k] &= P[X_n = j | X_{n+1} = i] \\ \frac{P[X_n = j, X_{n+1} = i, X_{n+2} = k]}{P[X_{n+1} = i, X_{n+2} = k]} &= \frac{P[X_n = j, X_{n+1} = i]}{P[X_{n+1} = i]} \end{aligned}$$

Interchanging the top right and bottom left terms:

$$\begin{aligned} \frac{P[X_n = j, X_{n+1} = i, X_{n+2} = k]}{P[X_n = j, X_{n+1} = i]} &= \frac{P[X_{n+1} = i, X_{n+2} = k]}{P[X_{n+1} = i]} \\ P[X_{n+2} = k | X_{n+1} = i, X_n = j] &= P[X_{n+2} = k | X_{n+1} = i] \end{aligned}$$

which is true by the Markov property of the original chain. We can establish $P[X_n = j | X_{n+1} = i, X_{n+2} = k, ...] = P[X_n = j | X_{n+1} = i]$ for arbitrarily far forward in time the same way.

We denote the reversed chain by $Q_{ij} = P[X_n = j | X_{n+1} = i]$.

What are the transition probabilities of the reversed chain? From the definition:

$$\begin{aligned} Q_{ij} &= P[X_n = j | X_{n+1} = i] \\ &= \frac{P[X_n = j, X_{n+1} = i]}{P[X_{n+1} = i]} \\ &= \frac{P[X_{n+1} = i | X_n = j] \, P[X_n = j]}{P[X_{n+1} = i]} \\ &= \frac{P_{ji} \, \pi_j}{\pi_i} \end{aligned}$$

and indeed we can interpret $Q_{ij}\pi_i = P_{ji}\pi_j$ as saying "the rate of transitions from i to j seen when watching the original chain is equal to the rate of transitions from j to i seen when watching the reversed chain." Note that in general, $P_{ij} \neq Q_{ji}$. The chain usually behaves differently when viewed in reverse!

One of the useful aspects of the reversed chain Q is that it is a *different* chain from P, but its steady-state probilities are the same as those of P. (The fraction of time the chain spends in any state doesn't change if we view the chain in the reverse direction.)

So, sometimes by thinking backwards, we can either explicitly construct Q, or guess at a possible Q, and use it to help find the π_is. The reason we can do this is the following:

Fact. Consider an irreducible DTMC with transition probabilities P_{ij}. If one can find positive numbers $\pi_i, i \geq 0$, summing to one, and a set of transition probabilities Q_{ij} such that

$$\pi_i P_{ij} = \pi_j Q_{ji}$$

for all i and j, then the Q_{ij} are the transition probabilites of the reversed chain and the π_i are the stationary probabilities of both the original and reversed chain.

Example: TCP's Window Size, Continued

In our previous study of TCP's window size, we were forced to study a finite chain in order to evaluate it numerically. Can we remove the upper limit on the window size, and study the chain analytically?

Let X_n equal i if the window size is i at time n. Let us denote by L the size of the window when a loss occurs. Then in our example, L has a geometric distribution with parameter p. That is,

$$P[L = i] = (1 - p)^{i-1} p.$$

In the forward direction, the chain always either increases by 1, or goes back to 1. So in the reverse direction, when the state is not 1, the chain always decreases by 1. That is:

$$Q_{i,i-1} = 1 \quad i > 1$$

What about state 1? There are transitions from state 1 to every other state, with probability $P[L = i]$ (Why?). So:

$$Q_{1,i} = (1-p)^{i-1} p \quad i \geq 1$$

To determine stationary probabilities, we must see if we can find $\{\pi_i\}$ such that

$$\pi_i P_{ij} = \pi_j Q_{ji}.$$

First, for $j = 1$:

$$\pi_i P_{i1} = \pi_1 Q_{1i}$$

which is

$$\begin{aligned}\pi_i p &= \pi_1 (1-p)^{i-1} p \\ \pi_i &= \pi_1 (1-p)^{i-1}\end{aligned}$$

To find π_1:

$$\begin{aligned}\sum_{i=1}^{\infty} \pi_i &= 1 \\ \sum_{i=1}^{\infty} \pi_1 (1-p)^{i-1} &= 1 \\ \sum_{i=1}^{\infty} (1-p)^{i-1} &= 1/\pi_1 \\ 1/p &= 1/\pi_1 \\ \pi_1 &= p\end{aligned}$$

So $\pi_i = p(1-p)^{i-1}$ for all i.

To finish the proof, we have to show that the remaining transition rates are also equal. That is:

$$\pi_i P_{i,i+1} = \pi_{i+1} Q_{i+1,i}$$

Which is true because:

$$p(1-p)^{i-1} \cdot (1-p) = p(1-p)^i \cdot 1$$

So what can we say about the average window size?

Can this be extended to a given maximum size N?

Exercise: Consider more general options for the distribution of L. In that case $p_i = P[L = i]/P[L \geq i]$. Note also that $\sum_{i=1}^{\infty} P[L \geq i] = E[L]$.

10.4 Markov Modulated Process

We can use DTMCs to model slightly more complicated situations as follows. Let us define a function $Y(s)$ for all states s of the DTMC.

Exercises

DTMCs

10-1. A search for Web pages containing a particular keyword returns 13 pages. The links between the pages are shown below. Compute the score that PAGERANK would assign to each page and rank the pages in order.

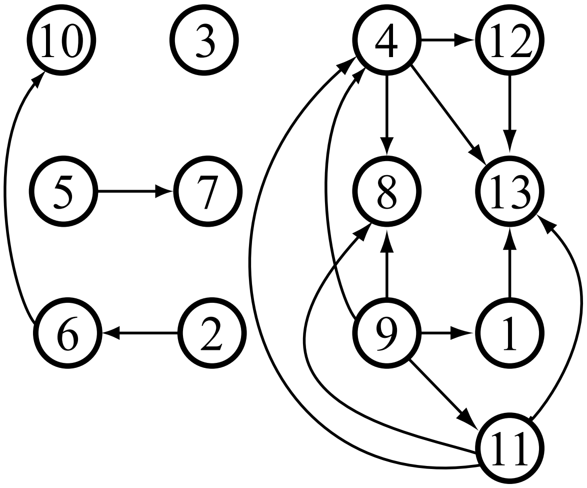


Figure 10.1: Pages for Exercise 1

10-2. *Describe the HITS/Clever model of Kleinberg. Apply it to the graph in the previous problem and state how the HITS algorithm would rank the pages. Compare to PageRank and discuss the differences.*

10-3. Consider the TCP modeling example in this chapter. Our modeling assumption that each window size has the same probability of being delivered intact $(1 - p)$ is obviously a bad one. A better model is that each packet has the same probability of being lost. Assume packet loss is independent, and that packets are lost with probability p.

(a) What probabilities should we assign to the edges of the Markov chain to incorporate the independent packet loss assumption?

(b) Construct a DTMC for this model and compare the it with the model developed in Exercise 6-1.

(c) Compare their predictions over the range p =0.5 to 0.005. How do the predictions differ? Explain why there is a difference.

(d) What changes might you make to improve the less-accurate model?

(e) Comment on which model lends more *insight* into the performance of the protocol.

Chapter 11

Continuous Time Markov Chains

11.1 Definition

We now consider a Markov process that evolves in continuous time.

We say that a process $\{X(t), t \geq 0\}$ is a *continuous time Markov chain* (CTMC) if for all $s, t \geq 0$ and nonnegative integers $i, j, x(u), 0 \geq u < s$,

$$P[X(t+s) = j | X(s) = i, X(u) = x(u), 0 \leq u < s] = P[X(t+s) = j | X(s) = i]$$

That is, this process has the Markovian property that the future depends only on the present state and is independent of the sequence of past states.

Let us assume that the process enters state i at some time. Let T_i be the time process stays in this state before transitioning to a new state (this is called the *sojourn time*. We can see from the CTMC definition that

$$P[T_i > s + t | T_i > s] = P[T_i > t].$$

Hence the random variable T_i is memoryless and so must be exponentially distributed (since the exponential distribution is the only continuous distribution that is memoryless).

Note that the process cannot “transition” directly from one state to the same state since that would be the same as simply remaining in that state.

Thus we can define the CTMC in another way. Let us call the expected time spent in state i $1/v_i$. Then

1. v_i is the rate of leaving state i, and the distribution of time spent in state i is exponential, that

is, it is given by

$$p(x) = v_i\, e^{-v_i x}.$$

2. when the process leaves state i, it goes next to state j with probability P_{ij}. As expected,

$$P_{ii} = 0,$$

and

$$\sum_j P_{ij} = 1.$$

Now it may seem that the CTMC has more parameters than the DTMC because it involves both the v_is and the P_{ij}s. However let us define

$$q_{ij} = v_i P_{ij}.$$

The interpretation of q_{ij} is as the rate of transitioning from state i to state j. Then the CTMC is completely described by the q_{ij}s alone, because:

$$v_i = \sum_j q_{ij},$$

and

$$P_{ij} = q_{ij}/v_i.$$

So we think of the CTMC as captured by the matrix Q, which is a matrix of *rates* of transitioning from i to j. Note that this is different from the DTMC case, in which P was a matrix of *probabilities.*

11.2 Recall!

Before proceeding with analysis of the CTMC, let us recall some important properties of the Exponential distribution and of Poisson Processes (from Module 4).

Comparing Two Exponential Random Variables: Suppose we have two independent random variables X_1 and X_2 that are each exponentially distributed, with respective parameters λ_1 and λ_2. Then $P[X_1 < X_2] = \frac{\lambda_1}{\lambda_1+\lambda_2}$.

Minimum of n Exponential Random Variables: Suppose that $X_1, X_2, ..., X_n$ are independent exponential random variables, with X_i having rate $\mu_i, i = 1, ..., n$. Now let us define a new random variable $Y = \text{minimum}(X_1, ..., X_n)$. Then Y is also an exponential random variable, with rate equal to the sum of the μ_i.

Superposition of Poisson Processes: Now let us consider two Poisson processes $N_1(t)$ having rate λ_1 and $N_2(t)$ having rate λ_2. Then the superposition of these two processes is a Poisson process with rate $\lambda_S = \lambda_1 + \lambda_2$.

Splitting of Poisson Processes: Probabilistically split a Poisson process into two substreams. Then if the splitting is done independently at each step, the two substreams are Poisson processes.

11.3 Limiting Probabilities

Does such a process have limiting probabilites like the discrete case? That is, is there a $\pi_j = \lim_{t\to\infty} P_{ij}(t)$ that is independent of i?

The answer is: yes, under the same conditions as for the DTMC case: when the associated chain is irreducible and ergodic. In this case, π_j exists and has the interpretation of the long-run proportion of time that the process is in state j.

To derive a set of equations for π_j, let us reason as follows. Assume that the CTMC has a steady-state occupancy distribution vector π, and consider an arbitrary state j. Now, over any interval of time $(0, t)$ the number of transitions into state j must be equal to within 1 of the number of transitions out of state j. Hence, in the long run, the rate of transitions into state j must equal the rate of transitions out of state j.

Now when a process is in state j, it leaves with rate v_j. Since π_j is the proportion of time that the process spends in state j, then

$$\pi_j v_j = \text{rate at which the process leaves state } j.$$

Similarly, when a process is in state k, it enters j at a rate q_{kj}. Since π_k is the proportion of time in state k, we can see that the rate at which transitions from k to j occur is $\pi_k q_{kj}$. So:

$$\sum_{k\neq j} \pi_k q_{kj} = \text{rate at which process enters state } j.$$

So we can derive an equation involving the π_js by setting the rate entering each state to the rate leaving each state. This equation is:

$$\pi_j v_j = \sum_{k\neq j} \pi_k q_{kj} \quad \text{for all states } j.$$

This is called a *flow balance* equation.

Now let us form the matrix Q as follows. For entries (i, j) with $i \neq j$, we set $Q_{ij} = q_{ij}$, that is, the off-diagonal entries are equal to the rate (when in state i) of transitioning from state i to state j. Let us set the diagonal entries $Q_{ii} = -v_i$, namely, the negative of the rate of leaving state i.

The structure of Q is then:

$$Q = \begin{bmatrix} -v_0 & q_{01} & q_{02} & \cdots & q_{0j} & \cdots \\ q_{10} & -v_1 & q_{12} & \cdots & q_{1j} & \cdots \\ q_{20} & q_{21} & -v_2 & \cdots & q_{2j} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ q_{i0} & q_{i1} & q_{i2} & \cdots & -v_i & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Then we can rearrage the previous equation as follows:

$$\sum_{k \neq j} (\pi_k q_{kj}) - \pi_j v_j = 0 \quad \text{for all states } j,$$

which in terms of our matrix Q can be expressed as:

$$\pi Q = 0,$$

which along with the normalizing equation

$$\sum_j \pi_j = 1$$

can be solved for the limiting probabilities.

11.3.1 Numerical View

Again, all of this can be cast in a numerical framework, this time based on $\pi Q = 0$ and $\sum_j \pi_j = 1$. We can construct any finite CTMC and express it as a matrix Q. This can then be solved using a numerical package. A completely linear algebraic approach to queueing theory is called the "matrix geometric" or "linear algebraic queueing theory" approach. A good book that takes this tack is [Lip92].

The equation $\pi Q = 0$ states that π is a vector in the nullspace of Q; the equation $\sum_j \pi_j = 1$ tells us which one it is. (Note that the rank of the nullspace of Q is 1, for the same reason that P's rank is $m - 1$ in the DTMC case.) To obtain π in, *e.g.,* matlab or octave:

```
PT = P'
S = null(PT)
pi = S(:,1)/sum(S(:,1))
```

In what follows we will primarily use analysis rather than numerical solution for two reasons: we will mainly look at infinite Markov chains, for which it is more accurate to work with analytic expressions; and we desire *insight* which is much more easily obtained from analytic expressions than numerical results.

However for most actual problems involving performance analysis via Markov Chains, the numerical approach is more practical. Furthermore, it is arguable whether equations or plots give more insight in any given situation. In the cases where we are using infinite CTMC models in a practical application, it is usually sufficient to use a large finite CTMC in place of the infinite CTMC. We will see that often in infinite CTMCs the probability of occupancy of most states is infinitesimally small and can be ignored in a practical setting.

11.4 Birth-Death Systems

Definition of a Birth-Death system: State of the system is represented by the number of people in the system at any time. Suppose that whenever there are n people in the system, new arrivals enter the system at the exponential rate λ_n and leave the system at the exponential rate μ_n. This is a *birth-death* system.

The key idea is that all state changes can only occur between adjacent states. That is, in passing from state n to state m, the system must pass at least once through all intermediate states.

This is a CTMC. What are the rates v_i and transition probabilities P_{ij}?

$$v_0 = \lambda_0$$

$$v_i = \lambda_i + \mu_i, \quad i > 0$$

$$P_{01} = 1$$

$$P_{i,i+1} = \frac{\lambda_i}{\lambda_i + \mu_i}, \quad i > 0$$

$$P_{i,i-1} = \frac{\mu_i}{\lambda_i + \mu_i}, \quad i > 0$$

Example. Consider a birth-death process for which

$$\mu_i = 0, \quad i \geq 0$$

$$\lambda_i = \lambda, \quad i \geq 0.$$

This is the Poisson Process.

Now let us solve the flow balance equation $\pi Q = 0$ for the Birth-Death system.

State	*Rate at which leave*	=	*Rate at which enter*
0	$\lambda_0\pi_0$	=	$\mu_1\pi_1$
1	$(\lambda_1+\mu_1)\pi_1$	=	$\mu_2\pi_2+\lambda_0\pi_0$
2	$(\lambda_2+\mu_2)\pi_2$	=	$\mu_3\pi_3+\lambda_1\pi_1$
$\vdots$	$\vdots$		$\vdots$
n	$(\lambda_n+\mu_n)\pi_n$	=	$\mu_{n+1}\pi_{n+1}+\lambda_{n-1}\pi_{n-1}$
$\vdots$	$\vdots$		$\vdots$

We find that we can solve this system in terms of π_0:

$$\begin{aligned}
\pi_1 &= \frac{\lambda_0}{\mu_1}\pi_0 \\
\pi_2 &= \frac{\lambda_0\lambda_1}{\mu_1\mu_2}\pi_0 \\
\pi_3 &= \frac{\lambda_0\lambda_1\lambda_2}{\mu_1\mu_2\mu_3}\pi_0 \\
&\vdots \\
\pi_n &= \frac{\lambda_0\lambda_1\cdots\lambda_{n-1}}{\mu_1\mu_2\cdots\mu_n}\pi_0
\end{aligned}$$

Now, to solve for π_0 the flow balance equations alone are not enough. However we can employ the fact that $\sum_n \pi_n = 1$ to obtain:

$$1 = \pi_0 + \pi_0 \sum_{n=1}^{\infty} \frac{\lambda_0\lambda_1\cdots\lambda_{n-1}}{\mu_1\mu_2\cdots\mu_n}$$

so:

$$\pi_0 = \frac{1}{1+\sum_{n=1}^{\infty}\frac{\lambda_0\lambda_1\cdots\lambda_{n-1}}{\mu_1\mu_2\cdots\mu_n}}$$

Note that it is necessary for $\sum_{n=1}^{\infty}\frac{\lambda_0\lambda_1\cdots\lambda_{n-1}}{\mu_1\mu_2\cdots\mu_n}$ to be finite in order for π_0 (and thus all limiting probabilities π_j) to exist.

Exercises

11-1. The Web Server at NetVapor, Inc. has two bugs. When these bugs are encountered, the Server crashes. Each time the server is rebooted, it remains up for an exponentially distributed time with rate λ. It then crashes, either due to Bug 1 or Bug 2. If it is due to Bug 1, it takes the system staff time to restart the system that is exponential with rate μ_1. If it crashes due to Bug 2, it takes time to restart that is exponential with rate μ_2. Each crash is due to Bug 1 with probability p and due to Bug 2 with probability $1 - p$.

(a) What proportion of time is the server down due to Bug 1?

(b) What proportion of time is the server down due to Bug 2?

(c) What proportion of time is the server running?

Part VII

Queues

Chapter 12

Queueing Theory

We now have the necessary tools to address the performance of queues.

12.1 Definitions

There are many different types of queues, so we taxonomize them using **Kendall notation**, which takes the form **A/B/C/D/E**, in which the various parts mean:

A	Distribution of time between arrivals
B	Distribution of time to service a customer
C	Number of Servers
D	Number of Customers that can wait in the queue (storage size)
E	Number of Customers that are available (in system; population size)

When specifying **A** and **B** (this distributions) we use the following symbols:

M Exponential (M is for “Markovian”)

D Deterministic (Constant)

E$_r$ Erlang-r

H$_k$ Hyperexponential-k

G General (no specific distribution; any distribution)

When the last two entries are ∞ we leave them off, so for M/M/1/∞/∞ we just write M/M/1.

Sometimes we also note the service discipline being used by the server, for example, FCFS, LCFS, Round-Robin, Random, Priority, etc.

12.1.1 System Utilization

Generally we will use λ to denote arrival rate (not necessarily corresponding to exponential interarrivals) and $\bar{x}$ to denote average service time.

We define **system utilization** ρ to be fraction of time the server is busy (or probability that the server is busy at any instant).

Now for the most fundamental fact about any computing system. Given a set of N interarrivals a_n and service times x_n, we have:

$$\begin{aligned}
\rho = P[\text{server is busy}] &= lim_{N\rightarrow\infty}\frac{\sum_{n=1}^{N} x_n}{\sum_{n=1}^{N} a_n} \\
&= lim_{N\rightarrow\infty}\frac{\frac{1}{N}\sum_{n=1}^{N} x_n}{\frac{1}{N}\sum_{n=1}^{N} a_n} \\
&= \frac{\bar{x}}{1/\lambda} \\
&= \lambda\bar{x}
\end{aligned}$$

That is, we have the basic identity $\rho = \lambda\bar{x}$.

12.1.2 Little’s Law

We will now derive the most useful fact in all of queueing theory.

For a system with arrival rate λ, and mean time in system T, the mean number in system N is given by:

$$N = \lambda T$$

This is true for any work-conserving queue, *regardless of the distribution of service times, the distribution of interarrival times, and the service discipline.* Thus this law has amazingly general applicability.

Here is an intuitive look at why this might be the case. Consider a drive-up Margarita stand. Let us observe a “typical” customer. The typical customer joins the end of the line, and in time T eventually makes it to the front of the line. During this time, λT customers have arrived, and λT customers have departed. Also, there are no customers in the system who arrived before our special customer. So clearly there were λT customers in the system when our customer arrived (and departed).

Of course, this simple example only seems to work when service is FCFS. The amazing thing about Little’s Law is that it works for *any* service discipline.

12.2 M/M/1

We begin by analyzing the simplest (and yet incredibly useful) queueing system, the M/M/1 queue.

We will model this queue with a CTMC, in which each state models the corresponding number of customers in the system. That is, state 0 corresponds to an empty system, state 1 to a single customer (in service), state 2 to a customer in service and one in the queue, etc.

This CTMC is clearly a birth-death system. In this system, $\lambda_n = \lambda$ and $\mu_n = \mu$. So the system is particularly simple:

$$\lambda_0 \lambda_1 \cdots \lambda_{n-1} = \lambda^n$$

and

$$\mu_1 \mu_2 \cdots \mu_n = \mu^n.$$

So using these facts and the Birth-Death equations we can solve for π_0:

$$\begin{aligned}\pi_0 &= \frac{1}{1 + \sum_{n=1}^{\infty} \frac{\lambda_0 \lambda_1 \cdots \lambda_{n-1}}{\mu_1 \mu_2 \cdots \mu_n}} \\ &= \frac{1}{1 + \sum_{n=1}^{\infty} \frac{\lambda^n}{\mu^n}} \\ &= \frac{1}{1 + \sum_{n=1}^{\infty} \rho^n}\end{aligned}$$

$$\begin{aligned} &= \frac{1}{1+\frac{\rho}{1-\rho}} \\ &= 1-\rho \end{aligned}$$

Question: Why should it be intuitively obvious that $\pi_0 = 1-\rho$?

Next, now that know π_0, we can solve for the other π_k's:

$$\begin{aligned} \pi_1 &= \frac{\lambda}{\mu}\pi_0 = \rho(1-\rho) \\ \pi_2 &= \frac{\lambda}{\mu}\pi_1 = \rho^2(1-\rho) \\ \pi_3 &= \frac{\lambda}{\mu}\pi_2 = \rho^3(1-\rho) \end{aligned}$$

... and so on. From the pattern here, we can see that $\pi_k = (1-\rho)\rho^k$. This is the geometric distribution with $p = 1-\rho$.

Well, actually, this is *almost* the geometric distribution. As we defined it in Module 3, the geometric distribution took on values $k > 0$. (Recall that the geometric distribution is $P[X = k] = p(1-p)^{k-1}$, with $k > 0$, and that $E[X] = 1/p$.) Here, we have $k \geq 0$. For that reason, the formula for π_k here differs from the one given in Module 3 by a factor of ρ.

At this point, we should pause and consider. We have found the steady-state distribution π of the underlying CTMC for the M/M/1 queue. We have all the information we need to obtain any performance measure we want at this point.

Question: What if ρ had been greater that 1? How would this have changed the analysis?

You can answer this question two ways: First, if $\rho > 1$, then $\sum_{n=1}^{\infty} \rho^n$ would be ∞, so π_0 would be zero. This leads to the conclusion that all π_k's are zero, an obvious impossibility. You can see that it was essential that $\rho < 1$ in order for this sum to be finite, given that Markov Chain itself is of infinite size.

Another way to look at it is to ask whether steady state even exists if $\rho > 1$. In fact, it does not: the CTMC would not be ergodic, because it would not be *recurrent*. (Why not?) This explains the impossibility in the last paragraph – the assumptions underlying the use of the CTMC are violated when $\rho > 1$.

Note how these observations depend on the infinite size of the CTMC. If the CTMC had been finite, then steady state could exist for $\rho > 1$.

12.2.1 Mean Number of Customers in System

The number of customers in the system when the CTMC is in state k, is k. So $E[k]$ is the expected number of customers in the system — including the customer in the queue.

What is the mean of this distribution (remembering that it's not exactly geometric, having $k \geq 0$)? Suppose we *did* have a RV X distributed geometrically with $p = 1 - \rho$. Then $\pi_k = P[X = (k+1)]$ for $k > 0$. So

$$\begin{aligned}
E[k] &= \sum_{k=0}^{\infty} k\,\pi_k \\
&= \sum_{k=0}^{\infty} k\,P[X = (k+1)] \\
&= \sum_{k=1}^{\infty} (k-1)\,P[X = k] \\
&= \sum_{k=1}^{\infty} k\,P[X = k] - \sum_{k=1}^{\infty} P[X = k] \\
&= E[X] - 1 \\
&= \frac{1}{1-\rho} - 1 \\
&= \frac{\rho}{1-\rho}
\end{aligned}$$

So $E[k] = \rho/(1-\rho)$. This is the *average number of customers in the system.*

12.2.2 Waiting Time

Immediately then by Little's Law we have the mean time spent in the system:

$$\bar{W} = \frac{N}{\lambda} = \frac{\rho/\lambda}{1-\rho} = \frac{\bar{x}}{1-\rho}$$

We can see this another way, using the "tagged customer" argument (covered in the next chapter). Since Poisson arrivals see time averages, the state of the system that an arriving customer sees is the same as the time average. So an arriving customer sees $\rho/(1-\rho)$ customers in the system when it arrives. Then the time the customer spends in the system, on average, is:

$$\text{time spent for customers in system on arrival} + \text{customer s own service time}$$

which is:

$$\bar{x}\,\frac{\rho}{1-\rho} + \bar{x} = \bar{x}\left(\frac{\rho}{1-\rho}\right) = \frac{\bar{x}}{1-\rho}.$$

12.2.3 Mean Number in Queue

Now, the probability a customer is in service can also be seen as the average number of customes in service (since this is a Bernoulli RV). So:

$$E[\text{number in queue}] = E[\text{number in system}] - E[\text{number in service}]$$

that is:

$$\bar{Q} = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho}$$

Question: What is the utilization of an M/M/1 queue that has 4 customers in the queue on average?

$$4 = \frac{\rho^2}{1-\rho}$$

$$\rho^2 + 4\rho - 4 = 0$$

$$\rho = 2\sqrt{2} - 2 \approx 0.82$$

12.3 What “Causes” Queueing?

Now, we start to ask a fundamental question: why does the queue build up as utilization increases? What causes queueing, anyway?

Consider a D/D/1 queue. Assume $\lambda < \bar{x}$ Then $\pi_1 = \lambda\bar{x} = \rho$. What is π_2? Clearly, zero. So $\pi_k = 0 \;\; k \geq 2$. So $\pi_0 = 1 - \pi_1 = 1 - \rho$. So *no* queueing ever occurs, regardless of the value of ρ!

12.4 M/M/1/K

A more realistic model of a real system would include the fact that queue space is finite: the M/M/1/K queue models this.

There are at most K customers in system: 1 in service, and K-1 waiting. When K are present, arriving customers are turned away, *i.e.,* lost.

Then:

$$\lambda_n = \begin{cases} \lambda & n = 0, 1, 2, \cdots, K-1 \\ 0 & n \geq K \end{cases}$$

and

$$\mu_n = \begin{cases} \mu & n = 1, 2, \cdots, K \\ 0 & n > K \end{cases}$$

Note that $\rho \neq \lambda\bar{x}$ because system is *not* work conserving. So instead we will define symbol $a = \lambda/\mu$.

This is a Birth-Death system. So, using the B-D formulas:

$$\pi_n = \left(\frac{\lambda}{\mu}\right)^n \pi_0 = a^n \pi_0$$

What is π_0?

$$1 = \sum_{n=0}^{K} \pi_n$$

So:

$$1 \;\; = \;\; \pi_0(1 + a + a^2 + \cdots + a^k)$$

$$\begin{aligned} &= \pi_0 \sum_{i=0}^{K} a^i \\ &= \pi_0 \frac{1-a^{k+1}}{1-a} \end{aligned}$$

So

$$\pi_0 = \frac{1-a}{1-a^{k+1}}$$

Note the above equations do not hold when $\lambda = \mu$. However using the B-D equations we can find that in this case, $\pi_n = 1/(K+1)$ for $n = 0, 1, ..., K$.

So

$$\pi_n = \begin{cases} \frac{a^n(1-a)}{1-a^{K+1}} & \lambda \neq \mu \\ \frac{1}{K+1} & \lambda = \mu \end{cases}$$

So, we have the complete state probability distribution for this queue. Now, so far we have not assumed that $\lambda < \mu$. Is this OK? In the case of the M/M/1 queue, we had to assume $\lambda < \mu$ in order for the underlying CTMC to be ergodic, because the system would not be positive recurrent if $\lambda > \mu$. (That is, the system state would tend to increase without limit.)

In the case of the M/M/1/K queue, the situation is different because the queue is finite. Remember that any finite CTMC must have at least one recurrent state (and therefore class). Since this chain is irreducible, the entire chain is recurrent and so ergodic. So a steady state exists for any values of λ and μ, even when $\lambda > \mu$.

Now: what is the rate at which customers are lost? This is the rate at which customers arrive when the system is full. So the loss rate is $\lambda\pi_k$.

What is the average number in system, $E[N]$?

$$\begin{aligned} E[N] &= \sum_{n=0}^{K} n\pi_n \\ &= \frac{1-a}{1-a^{K+1}} \sum_{n=1}^{K} na^n \\ &= \frac{a}{1-a} - \frac{(K+1)a^{K+1}}{1-a^{K+1}} \end{aligned}$$

So if $\lambda < \mu$, (that is, $a < 1$), then $E[N]$ is less than in the M/M/1 case, which makes sense, since some arriving customers were lost.

What about mean waiting time $E[W]$? The usual approach we have taken is to calculate the mean number of customers in the system and then use Little's Law. However we have to be careful here, because Little's Law only applies to a work-conserving system. Since some customers are turned away, this system is not work conserving.

However, we can imagine a work-conserving system "inside" this system. If we were to only consider those arriving customers who *do* receive service, and imagine the system as if only those customers arrived, the system performance (that is, $E[N]$) would be exactly the same. That system *would* be work-conserving, and so we could apply Little's Law.

To do this we need the arrival rate of customers that *do* receive service. This is clearly $\lambda(1 - \pi_K)$. So we can calculate the mean waiting time for this queue as:

$$E[W] = E[N]/(\lambda(1 - \pi_K))$$

Note how important π_K is. This value is called the *loss probability* of the system. It tells us how often the system loses incoming customers.

Example: Loss Rate at a Router. Assume we have a router serving packets that are arriving as a Poisson process, and assume that packet service time is exponentially distributed. We measure the router and note that the arrival rate times mean service time is 0.80 (where arrival rate includes also those packets that are dropped.) How many packet buffers should we allocate to limit the packet loss to 1/2%?

To solve this, we want the loss probability to be 0.005, and to solve for K. So:

$$\begin{aligned}
0.005 &= \frac{a^K(1-a)}{1-a^{K+1}} \\
&= \frac{(1-0.8)0.8^K}{1-0.8^{K+1}} \\
(0.8)^K &= 0.005/(1-0.8+0.8*0.005) = 0.0245
\end{aligned}$$

This yields $K \approx 16.6$, so buffer space for 17 packets will be sufficient to keep the loss rate below 1/2%.

12.5 M/M/c

Now we consider the case in which there are multiple servers, fed by a single queue. Some example applications include multiprocessors, cluster based servers, bank tellers.

The number of servers is c. When there are up to c customers in the system, there is no queue. If there are n customers in the system, and $n > c$, then $n - c$ customers are in the queue.

We will define $\rho = \frac{\lambda}{c\mu}$. As before, we will use the symbol a to mean λ/μ.

This is a birth-death system with *load-dependent service rate.*

In this system,

$$\lambda_n = \lambda$$

$$\mu_n = \begin{cases} n\mu & n = 1, 2, ..., c \\ c\mu & n \geq c \end{cases}$$

Plugging these into to our birth-death formulas yields:

$$\pi_n = \begin{cases} \frac{a^n}{n!}\pi_0 & n = 1, 2, ..., c \\ \rho^{n-c}\frac{a^c}{c!}\pi_0 & n > c \end{cases}$$

To solve for π_0:

$$\begin{aligned} 1 &= \pi_0 + \sum_{n=1}^{\infty} \pi_n \\ &= \pi_0(1 + \sum_{n=1}^{\infty} \pi_n/\pi_0) \\ \pi_0^{-1} &= 1 + \sum_{n=1}^{\infty} \pi_n/\pi_0 \\ &= 1 + a + \frac{a^2}{2!} + \frac{a^3}{3!} + \cdots + \frac{a^{c-1}}{(c-1)!} + \frac{a^c}{c!}(1 + \rho + \rho^2 + \cdots) \\ &= \sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!} \sum_{n=0}^{\infty} \rho^n \\ &= \sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!(1-\rho)} \end{aligned}$$

So

$$\pi_0 = \left(\sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!(1-\rho)} \right)^{-1}$$

Thus we have completely solved the M/M/c system. We can now proceed to calculate statistics of interest: mean waiting time, mean number of customers in system, mean queue length.

First, however we start with a simple question: What is the probability that an arriving customer will have to wait for service? This is called *Erlang's C formula:* $C[c, a]$.

$$\begin{aligned} C[c,a] &= P[N \geq c] \\ &= \sum_{n=c}^{\infty} \pi_n \\ &= 1 - \sum_{n=0}^{c-1} \pi_n \\ &= 1 - \pi_0 \sum_{n=0}^{c-1} \frac{a^n}{n!} \\ &= 1 - \frac{\sum_{n=0}^{c-1} \frac{a^n}{n!}}{\sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!(1-\rho)}} \\ &= \frac{\frac{a^c}{c!}}{(1-\rho)\sum_{n=0}^{c-1} \frac{a^n}{n!} + \frac{a^c}{c!}} \end{aligned}$$

Next we note that $P[Q=0] = 1 - C[c,a]$.

Now, to get started on performance measures, we start with the average number of customers in the queue:

$$\begin{aligned} E[Q] = \bar{Q} &= \sum_{n=c}^{\infty} (n-c)\pi_n \\ &= \sum_{k=0}^{\infty} k\pi_{c+k} \\ &= \pi_0 \frac{a^c}{c!} \sum_{k=0}^{\infty} k\rho^k \\ & \qquad \text{noting that: } \sum_{k=0}^{\infty} k\rho^k = \frac{\rho}{1-\rho} \text{ we get:} \\ &= \pi_0 \frac{a^c}{c!} \frac{\rho}{(1-\rho)^2} \\ &= C[c,a]\,\frac{\rho}{1-\rho} \end{aligned}$$

Now we can apply Little's law to the queue itself:

$$E[W_q] = E[Q]/\lambda = \frac{C[c,a]\bar{x}}{c(1-\rho)}$$

And, finally, we can observe that the total expected waiting time of a customer is the expected waiting time in the queue plus the expected service time:

$$E[W] = E[W_q] + E[W_s] = \frac{C[c,a]\bar{x}}{c(1-\rho)} + \bar{x}$$

Example: One queue or two? One server or two?. Consider the case in which we are going to upgrade an existing system, and have three options: we can double the service rate of the queue, or double the number of servers. Furthermore, if we double the number of servers, we could have both servers draw from a single, shared queue, or we could have the servers each draw from their own, private queue. Quantitatively speaking, which of the three options is best? (Ignoring issues of cost).

In this scenario, we have $\rho = \lambda/2\mu$, and $a = 2\rho$.

To start, let us find $C[2, a]$.

$$C[2, a] = \frac{\frac{a^2}{2!}}{(1-\rho)(1+a) + a^2/2!} = \frac{2\rho^2}{1+\rho}$$

Then

$$E[Q] = \frac{\rho C[2, a]}{1-\rho} = \frac{\rho\frac{2\rho^2}{1+\rho}}{1-\rho} = \frac{2\rho^3}{1-\rho^2}$$

So

$$E[N] = \frac{2\rho^3}{1-\rho^2} + 2\rho = \frac{2\rho}{1-\rho^2}$$

So

$$E[W] = \frac{E[N]}{\lambda} = \frac{\bar{x}}{1-\rho^2}$$

Now let us compare three cases. For clarity, in each case we will consider a total arrival rate of 2λ and a total service rate of 2μ.

Two M/M/1 queues. In this case, there are two separate queues, each with its own server, serving at rate μ. The customers arriving with rate 2λ are split equally (probabilistically) between the two queues, such that each queue has an input rate of λ. Thus we can analyze just one of the queues, as any particular customer only encounters one queue.

M/M/2. In this case, there are two servers, each serving with rate μ, drawing from a single shared queue. Customers arrive at rate 2λ.

Single M/M/1, fast server. In this case, customers arrive to a single server queue at rate 2λ and are served at rate 2μ.

Let us consider the corresponding expressions for mean waiting time in each case.

Case	$E[W]$
Two M/M/1 queues	$\frac{\bar{x}}{1-\rho}$
M/M/2 queue	$\frac{\bar{x}}{1-\rho^2}$
Faster M/M/1 queue	$\frac{\bar{x}}{2(1-\rho)}$

So we see that the difference between these cases depends on the load on the system. The worst case is Case 1, in which one server may be idle even when there is work to be done, because it cannot draw from the other server's queue. Case 2 is faster than Case 1 by a factor of $E[W_1]/E[W_2] = 1 + \rho$, and Case 3 is faster than Case 1 by a factor of 2. When ρ is small, Case 2 is about like Case 1; as ρ gets bigger (closer to 1), Case 2 becomes like Case 3.

Why is Case 2 worse than Case 3? Case 3 is faster than Case 2 by a factor of $E[W_2]/E[W_3] = 2/(1 + \rho)$. When there are many customers in the system, ($\rho \approx 1$) both systems are about the same, serving at rate 2μ. However, when there is only one customer in the system, Case 2 serves at rate μ while Case 3 serves at rate 2μ. So when system load is low ($\rho \approx 0$), Case 3 spends much of its time serving twice as fast as Case 2. This is the difference between the two.

The idea that a single queue is better than two queues is called multiplexing efficiency. In Case 2, two servers are "multiplexing" or sharing the queue, avoiding the problems of Case 1 in which work exists but cannot be serviced because it is in a queue that is not accessible to an idle server. This notion underlies the idea of packet-switched networks like the Internet, in which channels (queues) are *not* pre-allocated to connection. Each link in the network is free to serve a packet from *any* connection, leading to mulitplexing gains over the case in which each connection has pre-allocated resources.

12.6 M/M/∞

Let us now consider (mainly for later use, and for modeling insight) a system with an *infinite* number of servers. Each arriving customer gets their own server right away! Clearly there is no queue at all here. This is called a *pure delay* system. It is useful for modeling cases where there is a finite customer population, and a server for each customer. For example, consider a bunch of users sitting at terminals, reading email or chatting online. Each time a new message arrives, a user reads the message, then replies. Each user is having only one conversation, so there is never a message waiting.

This is a birth-death system. Let's construct the Markov Chain. Let arrival rate be λ and service time be exponential with mean $1/\mu$. Thus if there are k customers in the system, service rate is $k\mu$.

Under what conditions is the system ergodic? And, since mean delay in system is just equal to mean service time, the interesting question is: what is the mean number of customers in the system?

By B-D equations:

$$\pi_k = \frac{\lambda^k}{k!\mu^k}\pi_0$$

and

$$\pi_0 = \frac{1}{1 + \sum_{k=1}^{\infty}(1/k!)(\lambda/\mu)^k} = \frac{1}{\sum_{k=0}^{\infty}(1/k!)(\lambda/\mu)^k}$$

Now, using the identity:

$$\sum_{k=0}^{\infty}(1/k!)x^k = e^x$$

we obtain:

$$\pi_0 = e^{-\lambda/\mu}$$

So

$$\pi_k = \frac{(\lambda/\mu)^k}{k!}e^{-\lambda/\mu}$$

That is, the number of customers in the system is distributed according to the Poisson distribution, with expected value λ/μ.

Time in system: using Little's law $L = \lambda W$:

$$W = 1/\mu$$

of course.

12.7 M/E$_r$/1

We have already mentioned the question "what causes queueing?" The issue can be highlighted as follows. Consider a D/D/1 system with $\rho < 1$. How many customers are in the queue?

Now, as a comparison case we have the M/M/1 system in which we know that there are $\rho^2/(1-\rho)$ customers in the queue on average.

So queueing could be caused by variability in service times, or interarrival times, or both.

To understand and separate the effects of these two things, we'd like to study the M/D/1 queue. However it doesn't lend itself to simple representation as a CTMC since service times don't have the memoryless property.

However we can study the M/E$_r$/1 queue. This is useful because as we vary r, the coefficient of variation changes. The Central Limit Theorem tells us that for large r, the CV of the E$_r$ distribution goes like $1/\sqrt{r}$. In the limit of $r \to \infty$ the CV is zero, and the distribution is a constant (which is what we want to get to M/D/1).

Customers arrive with rate λ and have mean service time $1/\mu$. Thus the mean service time for a stage is $1/r\mu$ and the service rate for a stage is $r\mu$.

The E$_r$ distribution is not memoryless, but is the sum of memoryless "stages". So we will build a CTMC in which the states correspond to stages rather than customers in the system. We will think of each customer arrival as bringing r stages for service.

In this CTMC, transitions to the left are at rate $r\mu$. Transitions to the right jump r states and are at rate λ.

This is not a birth-death system, since there are jumps to states that are not immediately adjacent. So we can't use the B-D equations. However, as for all CTMCs, the key equations $\pi^T Q = 0$ and $\pi^T 1 = 1$ still hold. So we will build flow balance equations based on flow in to a state = flow out of a state.

These are:

$$\begin{aligned}
\lambda\pi_0 &= r\mu\pi_1 \\
(\lambda + r\mu)\pi_k &= r\mu\pi_{k+1} \quad 0 < k < r \\
(\lambda + r\mu)\pi_k &= r\mu\pi_{k+1} + \lambda\pi_{k-r} \quad k \geq r
\end{aligned}$$

Rearraging as usual, to solve in terms of π_0:

$$\begin{aligned}
\pi_1 &= \frac{\lambda}{r\mu}\pi_0 \\
\pi_{k+1} &= \frac{\lambda + r\mu}{r\mu}\pi_k \quad 0 < k < r \\
\pi_{k+1} &= \frac{\lambda + r\mu}{r\mu}\pi_k - \frac{\lambda}{r\mu}\pi_{k-r} \quad k \geq r
\end{aligned}$$

Note that the last equation is an r-th order difference equation. Such equations can be solved using advanced methods (the z-transform). Using those methods, we find the following:

$$E[K] = \frac{(r+1)\rho}{2(1-\rho)}$$

which tells us the expected number of stages in the system. This quantity is composed of two things: the expected number of stages in service, and the expected number of stages corresponding to customers in the queue. That is:

$$E[K] = E[C] + rN_q.$$

We can find $E[C]$ as:

$$E[C] = \sum_{c=1}^{r} c\rho/r = \rho\frac{r+1}{2}$$

So

$$N_q = \frac{E[K] - E[C]}{r} = \frac{\rho^2(r+1)}{2r(1-\rho)}.$$

This tells us what we wanted to know. When $r = 1$ service times are exponential and the expression agrees with the M/M/1 queue. When $r \to \infty$, service times are constant and the epression tells us:

$$\lim_{r\to\infty} \frac{\rho^2(r+1)}{2r(1-\rho)} = \frac{\rho^2}{2(1-\rho)}$$

In other words, half of the queueing in the M/M/1 case is due to variability in service times. So we can "assign blame" for queueing delays in the M/M/1 system equally to service time variability and interarrival time variability.

A general conclusion from this analysis is that "smoothing" out interarrival times can reduce delays. This done, *e.g.*, on some highways systems in California, and various versions of it are frequently proposed for dealing with congestion-related delays in Internet traffic.

12.8 M/G/1

We now turn to the study of queues with general service times: the M/G/1 system. In this case, service time can be drawn from any distribution.

The M/G/1 system is much harder to analyze than M/M/1, so instead of fully analyzing the system using CTMCs, we'll focus exclusively on the *average* behavior of the system (mean waiting time, mean queue length, etc.). In order to do this kind of analysis, we will use the PASTA principle.

Review “PASTA” from Module 4.

Review “Mean Residual Life” from Module 4.

M/G/1. Now we can analyze the M/G/1 queue. This is a much more general queue than any we have seen before. It allows any kind of service time distribution, so we won’t be able to solve this using a Markov Chain approach. As a result we won’t be able to derive the π_ns that would completely characterize the system. Instead we’ll only be able to solve for *average* statistics.

The first thing to realize is that if we are interested in the average case, then what an arriving customer experiences is the same as the time-average behavior of the system. This is because customers arrive according to a Poisson process, so we rely on the PASTA principle.

This means that we can consider what happens to a “typical” customer, who we will call the “tagged” customer. That is, we can calculate the average values a typical customer would see as equal to the time-average values of the system.

The time a customer spends in the queue has two parts: a) waiting time for the customer that is in service when the tagged customer arrives, and b) waiting time for the customers that are ahead in the queue.

Considering b) first, this can be written as

$$N_q E[x]$$

where N_q is the time-average number of customers in the queue. Each customer in the queue will take a $E[x] = \bar{x}$ time to be served (on average), so that much time must elapse before the “tagged” customer can start service.

Considering a), the expected service time remaining for the customer currently in service when the tagged customer arrives is just the mean residual life of the service time distribution, times the probability that a customer is in fact in service. That is,

$$(1-\rho)\cdot 0 + \rho(\text{mean residual life})$$

So

$$W_q = N_q E[x] + \rho \frac{E[x^2]}{2E[x]}$$

We can then apply Little’s Law to the queue to determine:

$$N_q = W_q \lambda$$

So

$$W_q = W_q \lambda E[x] + \rho \frac{E[x^2]}{2E[x]}$$

So

$$W_q - \lambda E[x] W_q = \rho \frac{E[x^2]}{2E[x]}$$

So

$$W_q(1-\rho) = \lambda E[x] \frac{E[x^2]}{2E[x]}$$

So

$$W_q = \frac{\lambda E[x^2]}{2(1-\rho)}$$

This is the famous Pollaczek-Kinchin (P-K) equation.

Then total waiting time in system is

$$E[W] = W_q + W_s = E[x] + \frac{\lambda E[x^2]}{2(1-\rho)}$$

and, by Little's Law (again!) we have

$$E[N] = \lambda E[W] = \rho + \frac{\lambda^2 E[x^2]}{2(1-\rho)}$$

Note that queueing time is proportional to the second moment of service time. That is, increased service time variability causes increased queueing even for constant ρ.

Example: (Check) M/M/1: Exponential Service Times. $E[X] = 1/\mu, E[X^2] = 2/\mu^2$.

Example: (Check) M/D/1: Constant Service Times. $E[X] = 1/\mu, E[X^2] = 1/\mu^2$.
In this case,

$$W_q = \frac{\lambda E[x^2]}{2(1-\rho)} = \frac{\lambda/\mu^2}{2(1-\rho)} = \frac{1}{2} E[x] \frac{\rho}{1-\rho}$$

This agrees with our previous analysis of the M/E$_r$/1 queue in the limit of large r. Once again, compare to queue in M/M/1:

$$W_q = \frac{\rho^2}{\lambda(1-\rho)} = E[x] \frac{\rho}{1-\rho}$$

Which "confirms" that half of the delay in the M/M/1 system can be said to be due to variability in service times.

Exercises

Queues

12-1. M/G/∞ queue. It is known that, regardless of the service time distribution, the number of customers in service in the M/G/∞ system has a Poisson distribution. In particular, if mean service time is $\bar{x}$ and arrival rate is λ, the number of customers in service N is given by:

$$P[N = k] = \frac{(\lambda\bar{x})^k}{k!}\, e^{-\lambda\bar{x}}.$$

Use this fact and the results of Exercise 6-3 to describe how you would initialize a perfect simulation of an M/G/∞ queue.

12-2. One queue or two. (Uses results from Exercises 4-2 and 6-8). You are in the grocery store and you notice that while most customers carts are nearly empty, a small number of customers have very full carts. You decide to model the number of items in a cart (and therefore the checkout time) by a Pareto random variable with parameters k and $\alpha > 2$.

(a) When you arrive at the checkout, you notice that there are two clerks, and each clerk is serving a customer. There are no customers in line and you can't see the contents of either customer's cart. Assume you have to pick one of the two lines to stand in, as is usual at the grocery store, and that you can't switch lines once you've chosen. Plot a graph of the time you expect to wait as a function of α between 2 and 3.

(b) Now assume that instead, the grocery store has organized things so that you can stand in a single line and wait for whichever clerk becomes free first. On the same graph as before, plot the amount of time you expect to wait.

(c) What if $1 < \alpha \leq 2$? Compare the two-line case to the single-line case qualitatively. How are they different?

12-3. Given an M/M/1 queue at steady state, with mean interarrival time of 0.25 seconds and mean service time of 0.20 seconds. answer the following questions:

(a) What is the probability that an arriving customer will be able to begin service immediately after arriving?

(b) What is the probability that an arriving customer will find more than 5 customers already in the system?

(c) What probability *distribution* would you use to model the waiting time for a customer that arrives to find exactly 5 customers already in the system?

12-4. Traffic to a network router arrives according to a Poisson process with rate of 14400 packets per second. The router can transmit 22 Mbits/second. Packet sizes are exponentially distributed with a mean of 176 bytes. Assume that a very large number of packet buffers is provided, and the packets are processed in FCFS order. Calculate:

- the average number of packets in the router,
- the average number of packets in buffers, and
- the average delay experienced by a packet passing through the router, and
- the fraction of time that the router is busy.

12-5. Using the analytic results we have developed for the $M/M/1$ queue, plot the predicted values of $\bar{W}$ and $\bar{Q}$ as a function of ρ for $0 < \rho < 1$. To do this, perform the calculations for finely spaced values of ρ so as to create visually "smooth" curves.

Plot the analytic results and the results you obtained via simulation in Exercise ** on the same set of axes. Does the analysis agree with simulation?

12-6. Requests arrive to the Microsomething Web Server as a Poisson process at a rate of one arrival per 20 seconds. The Web server can only accept a request if there are no more than two requests already present (this is bug that is expected to be fixed in the upcoming release, scheduled for late next year). Requests are served in first-come-first-served order (*i.e.,* one request is completely served before the next one begins to get service. Suppose that the amount of time required to service a request is exponentially distributed with mean of five seconds.

(a) What fraction of the server's time will it be busy satisfying requests?

(b) What fraction of Web requests are rejected?

(c) For requests that are accepted, how long must they wait on average before they are completed?

Now suppose that the amount of time required to service a request is exponentially distributed with mean of twenty seconds. Answer the same three questions as before.

12-7. On your first day as capacity planner at Random Engineering Ltd., you are presented with the following problem. Engineers at Random rely on an expensive visualization system to inspect the quality of their designs for next generation Internet-ready toasters (the system is called "ToasterViz"). Only one engineer can use the ToasterViz system at a time, and engineers wait in line for the system if it is busy when they need it.

It appears that engineers arrive at the system at a Poisson rate of λ arrivals per second, and the amount of time each engineer spends on the system is exponentially distributed with mean $1/\mu$ seconds.

Lately demand for the ToasterViz system has been growing. Your boss claims that engineers now have to wait too long and so is considering buying another system if it will make a big difference to the engineers. To best allocate her scarce computer budget, she poses a number of questions and asks for the solutions.

(a) Assume we decide to put the new system in the same room as the existing system. Now engineers will wait in one line that feeds both ToasterViz systems. In answering these questions let $\rho = \lambda/2\mu$ be the utilization of the entire system.

i. Using Kendall notation, what sort of queueing system is this?

ii. Draw a picture of the Markov Chain that models this system.

iii. Is this any special kind of Markov Chain? Give an expression for π_k in terms of π_0 and ρ.

iv. Solve for π_0 in terms of ρ. (Note that since there are two servers, $\pi_0 \neq 1 - \rho$).

v. Now solve for the steady state number of engineers in the system (*i.e.*, using the system, or waiting in line). In doing so, remember that

$$\sum_{k=0}^{\infty} kp^k = \frac{p}{(1-p)^2}.$$

Also solve for the average number of engineers in line.

vi. Now solve for the average time an engineer spends in the new system as a function of ρ and W_s ($W_s = 1/\mu$). Also solve for the amount of time an engineer waits in line.

vii. On one plot, show the average time an engineer will wait in line for the old system, and the new system, as a function of ρ for the old system. Note that for the old system, $\rho = \lambda/\mu$ while for the new system $\rho = \lambda/2\mu$. Your plot should in units of mean service time, *i.e.*, you should normalize the results by $1/\mu$ before plotting.

(b) Engineers claim that currently they wait in line 70 minutes on average, and that they use the system for 30 minutes on average.

i. What is the percent of time the current (old) system is being used?

ii. What will the utilization be of the new system?

iii. How long will engineers have to wait in line on average for the new system?

(c) The engineers make a suggestion: if there are going to be two systems, why not put them on opposite sides of the plant? Then, engineers could go to the system closest to their desk and they wouldn't have to walk as much. They figure that they can save 15 minutes of walking time this way for each time they use the system. Assume the placement is such that half of the engineers use each system.

i. On one plot, show the average time an engineer will wait in line under the centralized system and the decentralized system, as a function of ρ.

ii. Given current utilization, how long will engineers have to wait in line on average under this scheme?

iii. Under what circumstances it this a good idea? Consider the case in which utilization may increase over time.

12-8. Using the analytic results we have developed for the $M/G/1$ queue, plot the analytic results and the results you obtained via simulation in Exercise ** (for $M/E_k/1$ and $M/H_2/1$ queus) on the same set of axes. Does the analysis agree with simulation?

Chapter 13

Service Disciplines

It's considered good manners to let someone ahead of you in line if they have only a few items, and you have many; why? Now consider that many operating systems distinguish (or try to distinguish) between interactive and batch jobs; why?

These questions relate to *service order.* We will now consider what happens when customers are *not* served in FCFS order.

In our analysis so far, we have only considered FCFS service. However queueing theory can also shed a lot of light on the performance of other service disciplines.

For all of our examples we will study the M/G/1 queue. This is a very useful model, which applies (to varying degrees) to a wide range of systems and settings. In keeping with our M/G/1 analyses so far, we will focus only on average performance measures.

Note that in this chapter, waiting time will refer to waiting time *in queue,* ie., what we have been calling W_q elsewhere.

13.1 Priorities

The simplest case to consider in one in which each customer has some priority; lower number priorities are given preferential service.

We can distinguish two cases: preemptive and non-preemptive scheduling. In non-preemptive scheduling, each task, once begun, continuously receives service until it is completed. In preemptive scheduling, an incoming task of higher priority *suspends* the current task and immediately begins service. In what follows, we will always assume nonpreemptive scheduling. (It is only a little more difficult to analyze preemptive scheduling.)

We define a set of priorities $1, 2, ..., P$ with 1 the highest (more preferred) priority. We assume that customers of each priority form classes that have different arrival rates and service time distributions, so we have $\lambda_i, E[x_i],$ and $E[x_i^2], i = 1, 2, ..., P$.

Each class has an associated system utilization that describes how much of the service capacity of the system is being used for jobs of this class: $\rho_i = \lambda_i E[x_i], i = 1, 2, ..., P$.

As we did with M/G/1 FCFS, we will take advantage of the fact that Poisson arrivals see the same average behavior as the time average behavior of the system, so we will use a "tagged" customer argument as we did for M/G/1 FCFS. Furthermore, we will assume that each job class arrives as a Poisson process; then the merger of all job classes clearly is also a Poisson process.

As a warmup, we'll start by noting that highest priority customers are not affected by lower priority customers in the queue. In addition, an arriving priority 1 customer has to wait for a whatever customer is in service to finish service before it can begin service.

So priority 1 customers see an M/G/1 queue, with the residual lifetime of the customer in service equal to the average residual lifetime of the mixture of all jobs.

What about lower priority customers? In this case, there is an additional source of delays: customer of higher priority that arrive after the tagged customer, but before the tagged customer either starts service (non-preemptive) or leaves the system (preemptive).

We will now derive the mean waiting time in the queue for a customer of class m (W_m), under nonpreemptive scheduling.

From the arguments above it is clear that in general there are three components to W_m:

1. The mean residual lifetime of the customer currently in service.
2. Service times for customers of equal or higher priority than m in the queue. Note that for each class i, $N_i = \lambda_i W_i$ by Little's Law.
3. Service times for customers of higher priority than m that arrive during W_m. Note that the number of customers of class i arriving during W_m is $\lambda_i W_m$.

We will then set W_m to be the sum of these three components.

Component 1. We denote by W_0 the average delay to our tagged customer due to the customer in service. Since this system is nonpreemptive, the class of the tagged customer does not matter. We simply use the law of total expectation, over the set of all classes. The probability that the customer in service is of class i is ρ_i; and the mean residual lifetime of a customer of class i is $\frac{E[x_i^2]}{2E[x_i]}$:

$$W_0 = \sum_{i=1}^{P} \rho_i \frac{E[x_i^2]}{2E[x_i]}$$

$$= \quad 1/2 \sum_{i=1}^{P} \lambda_i E[x_i^2]$$

Component 2. There are N_i customers of class i in the queue when our tagged customer arrives, so the total number of customers in the queue that our tagged customer must wait for is:

$$\sum_{i=1}^{m} N_i = \sum_{i=1}^{m} \lambda_i W_i$$

and the time our tagged customer spends waiting for them is

$$\sum_{i=1}^{m} \lambda_i W_i E[x_i]$$

Component 3. Our customer is waiting in the queue for time W_m. So during this time, on average, $\lambda_i W_m$ customers of class i arrive. So the customers that arrive while our customer is in the queue (and so that jump ahead of our customer in line) is:

$$\sum_{i=1}^{m-1} \lambda_i W_m$$

and the time our customer must wait for them is

$$\sum_{i=1}^{m-1} \lambda_i W_m E[x_i]$$

Finally, we can develop an expression for *waiting time in the queue* for a customer of class m as:

$$\begin{aligned} W_m &= \text{time for (customer in service } + \text{ customers in queue at arrival } + \text{ higher pri arrivals)} \\ &= 1/2 \sum_{i=1}^{P} \lambda_i E[x_i^2] + \sum_{i=1}^{m} \lambda_i W_i E[x_i] + \sum_{i=1}^{m-1} \lambda_i W_m E[x_i] \end{aligned}$$

This equation is recursive in m. So to solve it, let us start from the base case, $m = 1$ (that is, the highest priority class). Also, for simplicity, we will continue to use W_0 for $1/2 \sum_{i=1}^{P} \lambda_i E[x_i^2]$.

$$\begin{aligned} W_1 &= W_0 + \rho_1 W_1 \\ &= \frac{W_0}{1 - \rho_1} \end{aligned}$$

Now we can solve for W_2:

$$\begin{aligned} W_2 &= W_0 + \rho_1 W_1 + \rho_2 W_2 + \rho_1 W_2 \\ W_2[1-\rho_1-\rho_2] &= W_0 + \rho_1 W_1 \\ &= W_0 + \frac{\rho_1 W_0}{1-\rho_1} \\ &= \frac{W_0}{1-\rho_1} \\ W_2 &= \frac{W_0}{[1-\rho_1-\rho_2][1-\rho_1]} \end{aligned}$$

Repeating this process for larger m, we find that the general solution is:

$$W_m = \frac{W_0}{\left(1-\sum_{i=1}^{m}\rho_i\right)\left(1-\sum_{i=1}^{m-1}\rho_i\right)} = \frac{1/2\sum_{i=1}^{P}\lambda_i E[x_i^2]}{\left(1-\sum_{i=1}^{m}\rho_i\right)\left(1-\sum_{i=1}^{m-1}\rho_i\right)}$$

For simplicity we denote $u_m = \sum_{i=1}^{m}\rho_i =$ the fraction of time the system works on jobs of size m or less. Then:

$$W_m = \frac{W_0}{(1-u_m)(1-u_{m-1})} = 1/2\frac{\sum_{i=1}^{P}\lambda_i E[x_i^2]}{(1-u_m)(1-u_{m-1})}$$

Thus we have the waiting time in the queue for each class. To get the average waiting time in the system over all classes, we simply use the law of total expectation. Let $p_i = \lambda_i/\sum_n \lambda_n$. Then:

$$W = \sum_{m=1}^{P} p_m(W_m + E[x_m])$$

Example. Consider a priority queueing system with 2 job classes:

Class	Arrival Rate	Service Time Distribution	
Priority 1	$\lambda_1 = 10$ tps	Erlang-2 with $E[X] = 0.04$ sec	$E[X^2] = 0.0024$ sec^2
Priority 2	$\lambda_2 = 0.5$ tps	Exponential with $E[X] = 0.1$ sec	(calculate!) $E[X^2] = 0.02$ sec^2

(N.B Var$_1$ = 0.0008, C_1^2 = 0.02; Var$_2$ = 0.01; C_2^2 = 1)
This system has $\rho = 0.45$.
Using the above formulas, we find that in this system:

$$\begin{aligned} W_1 &= 0.0283 \text{ seconds} \\ W_2 &= 0.0515 \text{ seconds} \\ W &= 0.0722 \text{ seconds} \end{aligned}$$

Now let us add a third job type, at a lower priority than the others:

Class	Arrival Rate	Service Time Distribution	
Priority 3	$\lambda_3 = 0.01$ tps	H_2 with $E[X] = 10.0$ sec	$E[X^2] = 500$ sec^2

(N.B. Variance = 400; $C^2 = 4$)
The new system has $\rho = 0.55$.
Then we find:

$$\begin{aligned} W_1 &= 4.195 \text{ seconds} \\ W_2 &= 7.627 \text{ seconds} \\ W_3 &= 10.17 \text{ seconds} \\ W &= 4.41 \text{ seconds} \end{aligned}$$

13.2 Size Based Service

Let's say we know amount of processing time required by a customer when she arrives. Then what is the best possible service discipline? Here, 'best' will mean the service discipline with lowest expected time in system.

We saw in the previous section that different classes of customers experienced different waiting times, and that the overall waiting time was a weighted average of what each class experienced. We can see that a good way to assign priorities is to do so in decreasing order of class processing-time mean. Doing this will tend to minimize the average u_i and therefore minimize W. In fact this can be shown to be the optimal assignment of priorities to classes (by an interchange argument).

This suggests that if we know the processing time required by customers, we should assign customers to classes according to their processing demands, with customers requiring less service being assigned to higher priority classes. Clearly, this should reduce the average time in system as compared to a discipline that does not use knowledge of processing demands (such as FCFS).

13.2.1 SJF

Assume that the processing demands of jobs has CDF $F(x), x > 0$.

So we are talking about a priority system based on ‘cutoff’ points

$$0 = d_0 < d_1 < ... < d_r$$

with d_r so large that $F(d_r) = 1$. In this system, a customer having processing demand P is assigned to class j such that $d_{j-1} < P \leq d_j$. This is a kind of size-based service discipline.

Based on results in last section, overall mean waiting time is

$$E[W] = \frac{\lambda E[X^2]}{2} \sum_{j=1}^{r} \frac{p_j}{(1 - u_{j-1})(1 - u_j)}$$

Clearly we can improve performance (mean time in system) by increasing the number of classes r, since that will make finer distinctions between customers of different sizes.

When we take the limit of this approach for $r \to \infty$ we get the *Shortest Job First*[1] discipline. This is a non-preemptive discipline in which, upon completion of each job, the next job chosen for service is the one with the least service demand.

Since we are working in the limit of $r \to \infty$ we need to switch to PDFs. We denote the pdf of service demand as $p(x)$. We assume the pdf is continuous, and get

$$E[W] = \frac{\lambda E[X^2]}{2} \int_{x=0}^{\infty} \frac{p(x)}{\left(1 - \lambda \int_0^x t\, p(t)\, dt\right)^2}\, dx$$

or, using F_w:

$$E[W] = \frac{\lambda E[X^2]}{2} \int_{x=0}^{\infty} \frac{p(x)}{\left(1 - \rho\, F_w(x)\right)^2}\, dx$$

For example. Working with uniform distribution, $p(x) = 1, 0 < x \leq 1$. For this distribution, $F_w(x) = x^2$, $E[X] = 1/2$, and $E[X^2] = 1/3$.

```
Munif.m:
function y = Munif(x,rho)
y =  1 ./ ((1 - rho .* (x.^2)).^2);
```

[1] or *Shortest Processing Time*

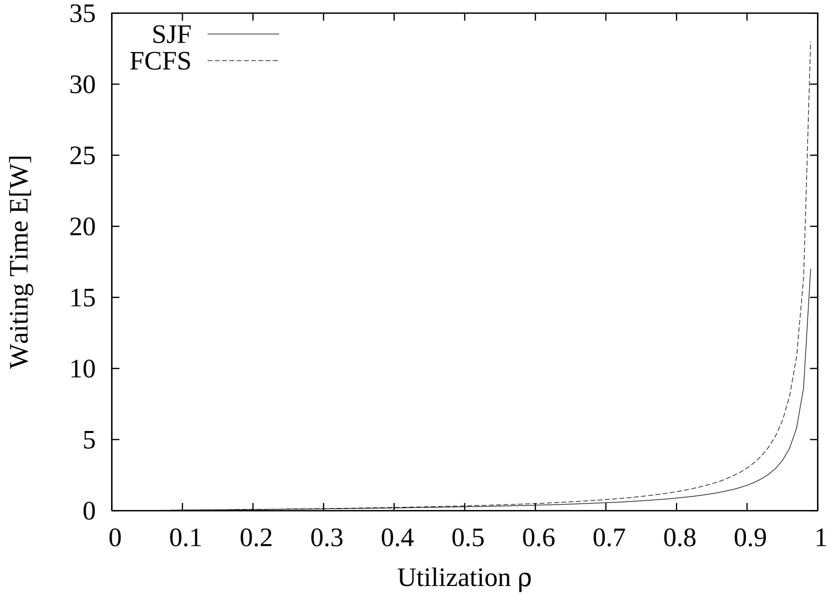


Figure 13.1: SJF and FCFS for Uniform Service Demands

```
WSJF.m:
function y = WSJF(fn,lo,hi,rho,firstmoment,secmoment)
y = (0.5 * rho * secmoment / firstmoment) * quad(fn,lo,hi,[],[],rho);
```

For $\rho = 0.7$:

```
WSJF(@Munif,0,1,0.7,1/2,1/3)
ans = 0.2788
```

The performance of SJF is compared to FCFS (where waiting time is $\lambda E[x^2]/2(1-\rho)$ in Figure 13.1.

Add another example here showing how results are different for a high-variance distribution.

13.2.2 SRPT

One problem with SJF is that when a “long” customer is being serviced, “short” customers that arrive have to wait. We can improve the mean time in system even more if we allow short jobs to pre-empt long jobs.

Two kinds of preemptive disciplines exist: preemptive resume and preemptive repeat. In the first case, when a preempted customer returns to service, work starts where it left off. That is, work performed is never re-done. In the second case, when a preempted customer returns to service, the customer's task must be restarted from the beginning.

Preemptive resume is more common in computer systems and so we will focus on that. The modifications for preemptive repeat are relatively straightforward.

We will still be concerned with the delay experienced by a customer, apart from its service demand. This includes waiting time, the time until a customer begins service. However, since a customer can be interrupted while in service, there is additional component of delay that is important. *Residence time* is the time between when a customer begins service and completes service. We are concerned with total delay, that is, waiting time plus residence time minus service time.

To achieve our goal of minimizing average response time, we must be careful about when we allow preemptions. In general we want the system to be working at all times with the goal that a customer can leave the system as soon as possible. Thus, if we implement a size-based preemption policy, for the customer in service it should be based on the *remaining demand* (rather than, say, the customer's original demand before it began service). Thus we want a preemptive resume policy in which a customer's class is based on its remaining processing time. This leads to the *Shortest Remaining Processing Time First* (SRPT) discipline.

As with SJF, we will approach this by defining a set of cutoff points

$$0 = d_0 < d_1 < ... < d_r$$

with d_r so large that $F(d_r) = 1$. Under SRPT, a customer having *remaining* processing demand P is assigned to class j such that $d_{j-1} < P \leq d_j$. Then a new-arriving customer will preempt the customer in service if the new customer has lower demand than the remaining processing for the customer in service.

Among preemptive policies making use of knowledge of service demand, SRPT is optimal. This can again be proved by an interchange argument.

We will consider mean waiting time and mean residence time separately.

Mean Waiting Time The class of a customer cannot change during its waiting time. Thus only difference between SJF and SRPT with respect to waiting time is that in some cases, the arriving job will preempt the job in service.

However, once a class k customer begins waiting, the waiting time is the same whether the system is preemptive or non-preemptive. That is because all jobs of class less than k must leave the system before the tagged job can run in both systems.

Recalling the expression for mean waiting time of customer of class k under a nonpreemptive policy:

$$E[W_k] = \frac{\frac{1}{2}\sum_{i=1}^{r} \lambda_i E[x_i^2]}{(1-u_{k-1})(1-u_k)}$$

the numerator represents the mean residual lifetime of the customer in service when a customer of class k arrives times the utilization of the system — in other words, the expected mean residual lifetime of the customer in service, as seen by an arriving customer of class k. Denoting this quantity by M_k, we can rearrange:

$$\begin{aligned} M_k &= \sum_{i=1}^{r} \rho_i \frac{E[x_i^2]}{2E[x_i]} \\ &= \frac{1}{2} \sum_{i=1}^{r} \lambda_i \, E[x_i] \, \frac{E[x_i^2]}{E[x_i]} \\ &= \frac{\lambda}{2} \sum_{i=1}^{r} p_i \, E[x_i] \, \frac{E[x_i^2]}{E[x_i]} \\ &= \frac{\lambda}{2} \sum_{i=1}^{r} p_i \, E[x_i^2] \\ &= \frac{\lambda}{2} \sum_{i=1}^{k-1} p_i \, E[x_i^2] + \frac{\lambda}{2} \sum_{i=k}^{r} p_i \, E[x_i^2] \\ &= \frac{\lambda}{2} \int_0^k p(x) \, E[x_i^2] \, dx + \frac{\lambda}{2} \int_k^\infty p(x) \, E[x_i^2] \, dx \end{aligned}$$

How should we adapt this expression for premption?

Now, as already mentioned the interference experienced by a customer of class k from customers of classes lower than k (higher priority customers) is the same for both preemptive and nonpreemptive policies. This is because in both cases, an customer of class k cannot go into service unless all customers of classes less than k, plus all previous class k customers, have left the system. So the first integral in the above equation is unchanged for SRPT.

However, since the system is preemptive, customers with remaining processing time greater than d_k will be preempted, and there will be no waiting. So the only time that a customer of class lower than k will not be preempted is when its remaining processing time has been reduced to d_k or less. That is, for a customer of class lower than k, only d_k of its processing time will not be preemptible by a class k customer. So from the standpoint of a customer of class k, it is as if all jobs of class lower than k arrived with demand d_k.

Thus, for these classes the second integral becomes:

$$\frac{\lambda}{2} \, d_k^2 \int_{d_k}^{\infty} p(x) \, dx = \frac{\lambda}{2} \, d_k^2 \, (1 - F(d_k))$$

Thus, for waiting time of a job of class k we obtain:

$$E[W_k] = \frac{\lambda \int_0^{d_k} x^2 \, p(x) \, dx + \lambda \, d_k^2 \, (1 - F(d_k))}{2(1 - \rho \, F_w(d_k))^2}$$

Mean Residence Time The second difference between preemptive and nonpreemptive policies is that under a preemptive policy, once a customer begins service there may be interruptions before the customer leaves the system. To compute the mean residence time, we consider how often a customer is interrupted in each class.

Assume that a customer requires p units of processing time while it is in class i before it can move to class $i-1$. While the job is in class i it is interrupted by all jobs of higher priority. So the time it spends in class i is $p/(1-u_{i-1})$.

The total residence time of a customer that arrives in class k is the sum of the residence times for $1 \leq i \leq k$. This is:

$$E[R_k] = \frac{E[x_k] - d_{k-1}}{1-u_{k-1}} + \sum_{i=1}^{k-1} \frac{d_i - d_{i-1}}{1-u_{i-1}}.$$

Taking the limit as the number of classes goes to ∞ and $d_k - d_{k-1}$ goes to zero yields:

$$E[R_k] = \int_0^{d_k} \frac{dx}{1-\lambda \int_0^x t\,p(t)\,dt} = \int_0^{d_k} \frac{dx}{1-\rho\,F_w(x)}$$

Putting it Together The resulting mean delay experienced by a random customer is the weighted average of the values for each class. When fully expanded, the result is:

$$E[W] = \int_0^{\infty} \underbrace{\frac{\lambda}{2} \frac{\int_0^x t^2\,p(t)\,dt + x^2\,(1-F(x))}{(1-\rho\,F_w(x))^2}}_{E[W_x]} + \underbrace{\int_0^x \frac{dt}{1-\rho\,F_w(t)}}_{E[R_x]}\, p(x)\,dx$$

Continuing our example for the uniform distribution, $F_w(x) = x^2$ and $\int_0^s x^2\,p(x)\,dx = s^3/3$....

13.3 Processor Sharing

A commonly used scheduling strategy is called *round-robin.* Round-robin works as follows: Customers are kept in a queue. The customer at the head of the queue begins service, and gets δ time units of service (or else completes service and leaves). If the customer has not left the system at the end of its quantum, it goes back to the rear of the queue. Then the customer that is now at the head of the queue enters service, and the process repeats.

Processor Sharing is an idealization of round-robin scheduling, in which the server switches between customers infinitely fast. Processor sharing is important because its properties are remarkably simple, and so it is a good approximation for the many situations in which round-robin is encountered.

Kleinrock's Theorem: Let $\delta \to 0$. (This is called a Processor Sharing system.) Then if there are k customers in the system on average, each customer gets $1/k$ of the server's capacity.

Furthermore, the departure process from the queue is Poisson, and the following equations hold:

$$\begin{aligned} \pi_k &= (1-\rho)\rho^k \\ N &= \frac{\rho}{1-\rho} \\ W &= \frac{E[x]}{1-\rho} \\ W[t] &= \frac{t}{1-\rho} \end{aligned}$$

That is, the average system characteristics are exactly the same as for M/M/1! (Note: $W[t]$ is the time spent in the system for a job of size t.)

The proof is based on constructing a discrete time Markov Chain, in which each time step corresponds to a single quantum of duration δ. Next we construct the reversed chain, in which arrivals correspond to departures in the forward chain. For the proper arrival process to the reversed chain, the reversed chain should have the same steady state distribution as the forward chain (as discussed in Section 10.3).

In fact, we show that if arrivals to the reversed chain are Bernoulli (which will become Poisson in the limit of small δ), then the reversed chain has the same steady state distribution as the forward chain. We can construct flow equations that relate π_k to π_{k-1} by setting equal corresponding flows in the forward and reversed chain, allowing us to solve for π_k. We confirm that these steady state probabilities are the same for both forward and reversed chains, proving that departures from the queue are Poisson and establishing the expression for π_k below in the limit of $\delta \to 0$.

Proof. Split time into timestep (or quantum) of size δ. Our DTMC will make a transition on each timestep.

The probability of an arrival in any timestep is $\lambda\delta$. Because δ will eventually go to 0, the probability of more than one arrival in a timestep is 0. When a new customer arrives, it immediately receives a quantum of service. The ith customer arrives with service demand X_i, which is an integer multiple of δ. All X_is are i.i.d. according to distribution $G(x)$.

The state of the DTMC will be a vector $\vec{s} = (m, z_1, z_2, ..., z_m)$. Customer z_1 is currently receiving service, z_2 is next to receive service, and so forth. On each transition, if no new customer arrives, then the zs rotate to the left.

Notation. $f(j) = P[X_i = j\delta].\bar{F}(j) = P[X_i > j\delta]$. We use $g(j)$ to denote he probability that a customer will depart after 1 more quantum, given that the customer has received j quanta so far; i.e.,

$$g(j) = f(j+1)/\bar{F}(j)$$

Forward Chain. First, we construct the forward chain to model round-robin. When $\vec{s} \neq \emptyset$, $\vec{s}$ can experience one of four things: (1) no arrival, no departure ($\vec{s} \to r(\vec{s})$); (2) one arrival, no departure ($\vec{s} \to a(\vec{s})$); (3) an arrival and a departure ($\vec{s}$ unchanged); (4) no arrival, just a departure ($\vec{s} \to d(\vec{s})$).

For $\vec{s} = (m, z_1, z_2, ..., z_m)$, these are defined as:

$$\begin{aligned} r(\vec{s}) &= (m, z_2, ..., z_m, z_1 + 1) \\ d(\vec{s}) &= (m - 1, z_2, ..., z_m) \\ a(\vec{s}) &= (m + 1, z_1, z_2, ..., z_m, 1) \end{aligned}$$

The case where $\vec{s}$ is unchanged occurs when a customer arrives with service demand of δ, recieves service, and leaves.

When $\vec{s} = \emptyset$, there are three possibilities:

$$\begin{aligned} \vec{s} &\to a(\vec{s}) = (1, 1) \quad \text{an arrival} \\ \vec{s} &\to \emptyset \quad \text{an arrival and a departure} \\ \vec{s} &\to \emptyset \quad \text{no arrival} \end{aligned}$$

Assuming $\vec{s} \neq \emptyset$, the probabilities of the forward chain are as follows:

$$\begin{aligned} P_{s,r(s)} &= (1 - \lambda\delta)(1 - g(z_1)) \\ P_{s,d(s)} &= (1 - \lambda\delta)g(z_1) \\ P_{s,a(s)} &= \lambda\delta(1 - f(1)) \\ P_{s,s} &= \lambda\delta f(1) \end{aligned}$$

These compound probabilities arise because $g(z_1)$ is the probability that the current customer departs, $f(1)$ is the probability that an arriving customer has service demand 1, and $\lambda\delta$ is the probability that a customer arrives.

When $\vec{s} = \emptyset$,

$$\begin{aligned} P_{s,a(s)} &= \lambda\delta(1 - f(1)) \\ P_{s,s} &= \lambda\delta f(1) + (1 - \lambda\delta) \end{aligned}$$

Reversed Chain. Now we construct the reversed chain. The key here is that we *guess* the transition probabilities of the reversed chain, and we *guess* the nature of the departure process, which is the arrival process for the reversed chain. We will guess that the reversed chain has Bernoulli arrivals with probability $\lambda\delta$ of having an arrival during a timestep.

Here are our guesses:

- $P^*_{r(s),s}$ corresponds to the case where there is no arrival in the backward chain. So the forward chain just served the customer at the head of the queue. So the probability that, if the chain is in a given state, that it came from the state corresponding to servicing a customer, is $1 - \lambda\delta$.

$$P^*_{r(s),s} = 1 - \lambda\delta.$$

- $P^*_{d(s),s}$
- $P^*_{a(s),s}$
- $P^*_{s,s}$

Example, Continued. Let's continue to consider the three-class case. In this system, $\rho = 0.55$ and $E[x] = 0.07105$ so $W = 0.157$ sec.
Also note: $W[1] = 0.0889$ sec; $W[2] = 0.222$ sec; $W[3] = 22.22$ sec.
Also, for comparison, let us calculate W under FCFS.

$$E[X^2] = \sum_i p_i E[X_i^2] = 0.479$$

and

$$E[X] = \sum_i p_i E[X_i] = 0.0523$$

So

$$W = E[x] + \frac{\lambda E[x^2]}{2(1-\rho)} = 0.0523 + \frac{10.51 \times 0.479}{2(1-.55)} = 5.64 \text{ seconds}$$

So we have:

	FCFS	Priorities	Processor Sharing
$W_1 + E[X_1]$	5.63 sec	4.23 sec	0.0889 sec
$W_2 + E[X_2]$	5.69 sec	7.72 sec	0.222 sec
$W_3 + E[X_3]$	15.59 sec	20.2 sec	22.22 sec
W	5.64 sec	4.41 sec	0.157 sec

Note that Priority based queueing does better for Class 1 than FCFS, and that mean waiting time for Priority queueing is better than for FCFS. Mean time will always be improved over FCFS if priorities are assigned in order of increasing job size. (Why?)

Note that Class 3 does slightly worse under PS than under Priority queueing, while the other two classes do better. This is an example of Kleinrock's Conservation Law: improvement in waiting time for one set of customers must come at the expense of another set of customers. This rule is usually written:

$$\sum_i \rho_i W_i = \text{a constant, regardless of schedule}$$

You may also want to consider what would have happened if priorities had been assigned in reverse order above. Where would that policy fit in the table?

Finally, it is often felt that this kind of priority based scheduling is “unfair” because it penalizes the large customers to benefit the small customers. However, this comparison is usually implicitly made with respect to FCFS. (Somewhere along the line many of us were taught that FCFS is “fair”!). However, when one asks what “fairness” really means, it is not so clear. If job size is known, it’s not clear why FCFS is considered fair. Perhaps a better definition of “fair” is in terms of *slowdown,* which is defined as

$$S[t] = W[t]/t$$

wher $S[t]$ is slowdown of a job of size t, and $W[t]$ is the total time in system for a job of size t. If all jobs had equal slowdown, this would mean that large jobs and small jobs both a delayed in constant proportion to their service demand.

Let us calculate slowdown for the three scheduling schemes above:

	FCFS	Priorities	Processor Sharing
S_1	140	105	2.22
S_2	56.9	77.2	2.22
S_3	1.55	2.02	2.22
S	135	103	2.22

This gives us a very different picture of “fairness.” In this light, Priority scheduling is more fair than FCFS, because FCFS dramatically *penalizes* small jobs. Most fair of all (“perfectly fair”) is PS, which slows all jobs sizes down equally. Perhaps this is why we allow shoppers with nearly empty shopping baskets ahead of us in line at the grocery store!

Exercises

Queues

13-1. No exercises as of yet.

Appendix

Student's t Distribution

For n degrees of freedom. The values below are one-sided. Thus for a 90% confidence interval, you want to use the value in the 95% column below.

n	90%	95%	97.5%	99.5%
1	3.07766	6.31371	12.7062	63.656
2	1.88562	2.91999	4.30265	9.92482
3	1.63774	2.35336	3.18243	5.84089
4	1.53321	2.13185	2.77644	4.60393
5	1.47588	2.01505	2.57058	4.03212
10	1.37218	1.81246	2.22814	3.16922
30	1.31042	1.69726	2.04227	2.74999

Bibliography

[ABCdO96] Virgílio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, pages 92–103, December 1996. Won award from Compaq/Brazil for one of ten best CS papers in 1996. http://www.uniemp.br/uniemp/compaq.

[AFT98] Robert J. Adler, Raisa E. Feldman, and Murad S. Taqqu, editors. *A Practical Guide To Heavy Tails*. Chapman and Hall, New York, 1998.

[AJB99] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[All90] Arnold O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Computer Science and Scientific Computing. Academic Press, Inc., 2nd edition, 1990.

[AW97] Martin F. Arlitt and Carey L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.

[BC98] Paul Barford and Mark E. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of Performance '98/SIGMETRICS '98*, pages 151–160, July 1998. Software for Surge is available from Mark Crovella's home page.

[BC99] Paul Barford and Mark E. Crovella. A performance evaluation of hyper text transfer protocols. In *Proceedings of ACM SIGMETRICS '99*, pages 188–197, May 1999.

[BCF$^+$99] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of INFOCOM '99*, pages 126–134, 1999.

[BKM$^+$00] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web: experiments and models. In *Proceedings the Ninth World Wide Web Conference (WWW9)*, 2000.

[Bou04] J.-Y. Le Boudec. Understanding the simulation of mobility models with Palm calculus. Technical Report IC/2004/53, EPFL, 2004.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[BV05] J.-Y. Le Boudec and M. Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *Proceedings of IEEE INFOCOM Conference*, 2005.

[CB97] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, December 1997. Revised and substantially corrected version of [Crovella and Bestavros, 1996].

[CBC95] Carlos A. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of WWW client-based traces. Technical Report TR-95-010, Boston University Department of Computer Science, April 1995. Revised July 18, 1995.

[CL99] Mark E. Crovella and Lester Lipsky. Simulations with heavy-tailed workloads. In Kihong Park and Walter Willinger, editors, *Self-Similar Network Traffic and Performance Evaluation*. Wiley / Wiley Interscience, New York, 1999. Slightly revised version of [Crovella and Lipsky, 1997].

[CMM67] Richard W. Conway, William L. Maxwell, and Louis W. Miller. *Theory of Scheduling*. Addison-Wesley Publishing Company, 1967.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of SIGCOMM '99*, 1999.

[FW97] Anja Feldmann and Ward Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. In *Proceedings of IEEE INFOCOM'97*, pages 1098–1116, April 1997.

[GK98] Charles M. Goldie and Claudia Kluppelberg. Subexponential distributions. In Robert J. Adler, Raisa E. Feldman, and Murad S. Taqqu, editors, *A Practical Guide To Heavy Tails*, pages 435–460. Chapman & Hall, New York, 1998.

[Gla94] Steven Glassman. A caching relay for the World Wide Web. In *Proceedings of the First International World Wide Web Conference*, pages 69–76, 1994.

[GLR92] Sharad Garg, Lester Lipsky, and Maryann Robbert. The effect of power-tail distributions on the behavior of time sharing computer systems. In *1992 ACM Symposium on Applied Computing*, Kansas City, MO, March 1992.

[GMR$^+$98] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, and E. L. Miller. Self-similarity in file systems. In *Proceedings of SIGMETRICS '98*, pages 141–150, 1998.

[HBD97] M. Harchol-Balter and A. Downey. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, 1997.

[Irl94] Gordon Irlam. Unix file size survey - 1993. Available at `http://www.base.com/gordoni/ufs93.html`, September 1994.

[Kle76] Leonard Kleinrock. *Queueing Systems*, volume II. Computer Applications. John Wiley & Sons, 1976.

[Lip92] L. Lipsky. *Queueing Theory: A Linear Algebraic Approach*. MacMillan, 1992.

[LM05] Amy N. Langville and Carl D. Meyer. Deeper inside PageRank. *Internet Mathematics*, 2005.

[LO86] W. E. Leland and T. J. Ott. Load-balancing heuristics and process behavior. In *Proceedings of Performance and ACM Sigmetrics*, pages 54–69, 1986.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[Mat89] Georges Matheron. *Estimating and Choosing: An Essay on Probability in Practice*. Springer-Verlag, 1989.

[Mol89] Michael Molloy. *Fundamentals of Peformance Modeling*. Macmillan, 1989.

[MSAHB04] David T. McWherter, Bianca Schroeder, Anastassia Ailamaki, and Mor Harchol-Balter. Improving preemptive prioritization via statistical characterization of OLTP locking. In *International Conference on Data Engineering (ICDE)*, 2004.

[NHM$^+$98] Norifumi Nishikawa, Takafumi Hosokawa, Yasuhide Mori, Kenichi Yoshida, and Hiroshi Tsuji. Memory-based architecture for distributed WWW caching proxy. *Computer Networks and ISDN Systems*, 30:205–214, 1998.

[PA96] David L. Peterson and David B. Adams. Fractal patterns in DASD I/O traffic. In *CMG Proceedings*, December 1996.

[Pax94] Vern Paxson. Empirically-derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, August 1994.

[Pet96] David L. Peterson. Data center I/O patterns and power laws. In *CMG Proceedings*, December 1996.

[PG95] D. Peterson and R. Grossman. Power laws in large shop DASD I/O activity. In *CMG Proceedings*, pages 822–833, December 1995.

[Ros02] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, eighth edition, 2002.

[Rou05] Matthew Roughan. Fundamental bounds on the accuracy of network performance measurements. In *Proceedings of ACM SIGMETRICS*, June 2005.

[Roz69] Y. A. Rozanov. *Probability Theory: A Concise Course*. Dover Publications, Inc., 1969.

[SF03] Kavé Salamatian and Serge Fdida. A framework for interpreting measurement over Internet. In *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, pages 87–94, August 2003.

[Zip49] G. K. Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, Cambridge, MA, 1949.