1 Simulating a single-server queueing model

Here we introduce a single-server queueing model, and how to simulate it. A good example to think about for intuition is an ATM machine. We view the machine as a "server" that serves customers one at a time. The customers arrive randomly over time and wait in a queue (line), and upon beginning service, each customer spends a random amount of time in service before departing.

1.1 FIFO single-server model

There is one server (clerk, machine), behind which forms a queue (line) for arriving customers to wait in. The n^{th} customer is denoted by C_n and arrives at time t_n, where

$$0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots,$$

with $\lim_{n\to\infty} t_n = \infty$.

$T_n \stackrel{\text{def}}{=} t_{n+1} - t_n$ denotes the n^{th} interarrival time, the length of time between arrival of the successive customers C_n and C_{n+1}.

C_n requires a service time of length S_n, which is the length of time C_n spends in service with the server. We assume that the server processes service times at rate 1, meaning that, for example, if C_n enters service now with $S_n = 6$, then 4 units of time later there are 2 units of service time remaining to process. D_n, called the *delay* of C_n, denotes the length of time that C_n waits in the queue (line) *before entering service*; if C_n arrives finding the system empty, then C_n enters service immediately and so $D_n = 0$. Summarizing: C_n arrives at time t_n, waits in the queue for D_n units of time, then spends S_n units of time with the server before departing at time $t_n^d = t_n + D_n + S_n$, the n^{th} departure time. We are inherently assuming here that customers join the end of the queue upon arrival an enter service one at a time, and this is known as *first-in-first-out* (FIFO). But other service disciplines are useful in other applications, such as in computer processing, where *processor sharing* (PS) might be employed: If there are $k \geq 1$ "jobs" in the system, they all are in service together, but each is served at rate $1/k$. We will discuss disciplines later on, so for now we assume FIFO.

FIFO delay has a nice recursive property:

Proposition 1.1

$$D_{n+1} = (D_n + S_n - T_n)^+, \ n \geq 0, \tag{1}$$

where $x^+ = \max\{x, 0\}$ *denotes the positive part of a number* x.

A proof of the above recursion is supplied by introducing *workload* $V(t)$ = the sum of all remaining service times in the system at time t. For example, suppose at time t there are 3 customers in the system. The one in service has some remaining service time S_r, and the other two waiting in queue have service times $\tilde{S}_1$ and $\tilde{S}_2$ respectively (say). Then $V(t) = S_r + \tilde{S}_1 + \tilde{S}_2$. A little thought reveals that in fact $V(t)$ is the delay, D_n, that C_n would experience if C_n arrived at time t: C_n would have to wait in queue for $V(t)$ units of time before entering service. Thus, in general D_n can be rewritten as $D_n = V(t_n-)$, the work found in system when C_n arrives at time t_n. We use "t_n-" to stress that the service time S_n of C_n is not included into the workload *found* by C_n; but right after C_n arrives and joins the queue the S_n is included into the workload. So: $D_n = V(t_n-)$ and $V(t_n+) = V(t_n) = D_n + S_n$. In words, then, the recursion for delay in Proposition 1.1 can be stated as: The work found by C_{n+1} is equal to the work found

by C_n, plus C_n's service time added, S_n, but minus the amount of work processed during the interarrival time T_n. Since work is processed at rate 1, this amount processed is at most T_n; the workload would become empty if $V(t_n-)+S_n < T_n$; hence the need for taking the positive part.

A nice way to think about workload is to imagine that our system is a reservoir filled with water (work), and that the water is drained off at constant rate 1. $V(t)$ denotes the water level at time t, and at time t_n, an amount S_n of water is dumped into the reservoir by C_n, causing the workload to jump upwards by the amount S_n. So, $V'(t) = -1$, whenever $V(t) > 0$, and $V(t_n+) - V(t_n-) = S_n$, for all $n \geq 1$. With this view, we easily conclude that

$$V(t_{n+1}-) = (V(t_n-) + S_n - T_n)^+,$$

which is precisely the recursion for delay in Proposition 1.1: The water level at time $t_{n+1}-$ is equal to the water level at time t_n-, plus the water added, S_n, but minus the amount of water drained during the interarrival time T_n. Since water is drained at rate 1, this amount drained is at most T_n; the reservoir would become empty if $V(t_n-)+S_n < T_n$; hence the need for taking the positive part.

1.2 Simulating the FIFO GI/GI/1 queue

The delay recursion (1) can be readily used to simulate the delay sequence $\{D_n : n \geq 0\}$. We of course must make some assumptions about the sequence of service times $\{S_n\}$ and interarrival times $\{T_n\}$. In applications these would be random variables, so we shall assume that $\{S_n\}$ is an iid sequence with common distribution denoted by $G(x) = P(S \leq x),\ x \geq 0$ and independently $\{T_n\}$ is an iid sequence with common distribution denoted by $A(x) = P(T \leq x),\ x \geq 0$. In practice, one would go out and collect data to figure out what G and A should be; for example, maybe the arrival process is a Poisson process at rate λ, meaning that $A(x) = 1 - e^{-\lambda x}$, and maybe the service times are uniformly distributed over some interval (a, b), meaning that $G(x) = (x-a)/(b-a),\ x \in (a, b)$. In any case, this model is known as the *GI/GI/1* queue, where the first GI refers to the interarrival times having a general distribution (A here) and being iid, and the second GI independently refers to the service times having a general distribution (G here) and being iid.

Let us assume that the inverse functions $G^{-1}(y)$ and $A^{-1}(y)$ are known so that the inverse transform method can be employed to generate our two sequences of rvs $\{S_n\}$ and $\{T_n\}$. Letting $\{U_n : n \geq 0\}$ and $\{V_n : n \geq 0\}$ denote two independent sequences of iid uniform$(0, 1)$ rvs generated by your computer, we can generate $S_n = G^{-1}(U_n)$ and $T_n = A^{-1}(V_n),\ n \geq 0$ to use in the recursion. To be precise, let's start with $D_0 = 0$. Then $D_1 = (S_0 - T_0)^+ = (G^{-1}(U_0) - A^{-1}(V_0))^+$ can be simulated. Now that we have D_1, we then can simulate D_2 via $D_2 = (D_1 + G^{-1}(U_1) - A^{-1}(V_1))^+$. Continuing out to any desired n, we can obtain $D_n = (D_{n-1} + G^{-1}(U_{n-1}) - A^{-1}(V_{n-1}))^+$.

1.2.1 Long-run (infinite horizon) simulation: estimating average delay

One measure of performance of use in applications is the average delay over all customers

$$d = \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} D_j.$$

In general, computing this exactly is not possible, so we can approximate it by using as our estimate

$$\frac{1}{n} \sum_{j=1}^{n} D_j,$$

where n is chosen to be large (10,000 say). This estimate requires only one simulation run of the n rvs $D_1, \ldots, D_n$, that is, your computer would be asked to do this just once. We refer to this as long-run simulation since we are trying to estimate something (a limit in time) that requires all of the infinite future.

As another example, we might wish to estimate the proportion of customers who have a delay exceeding some given (high) fixed value x;

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} I\{D_j > x\},$$

where $I\{A\}$ denotes the indicator of an event A; $I\{A\} = 1$ if A occurs, 0 otherwise.

1.2.2 Finite-horizon simulation: Monte Carlo

Instead of computing average delay over all customers, suppose we are only interested in the expected delay of the 5^{th} one. That is, we want to estimate $E(D_5)$, an expected value. In this case we would ask the computer to simulate n iid copies of D_5, where n is large and fixed, denoted by $X_1, \ldots, X_n$, and use as our estimate the empirical average

$$\frac{1}{n} \sum_{j=1}^{n} X_i \tag{2}$$

Here, since the X_i are iid, we justify our estimate via the strong law of large numbers which asserts that wp1, we get the exact answer $E(D_5)$ as $n \to \infty$. Each independent replication (copy) X_i of D_5 requires a new independent simulation run. The first run would be carried out via $D_1 = (S_0 - T_0)^+ = (G^{-1}(U_0) - A^{-1}(V_0))^+$, $D_2 = (D_1 + G^{-1}(U_1) - A^{-1}(V_1))^+$, up to the desired $X_1 = D_5 = (D_4 + G^{-1}(U_4) - A^{-1}(V_4))^+$. We then, using new uniforms, independently do this all over again to obtain another copy X_2 and so on, until we have a total of n such independent copies. Then we use the estimate in (2).

Another example is estimating the average delay over a given fixed number of customers N ;

$$E\Big[\frac{1}{N} \sum_{j=1}^{N} D_j\Big].$$

Here, each copy X_i would be a copy of

$$\frac{1}{N} \sum_{j=1}^{N} D_j,$$

and would require its own run. Each run would require the generation of the fixed number N of the D_j's.

One might instead wish to simulate only up to a a *fixed time* T. For example, a web site offering some special service might open at midnight and then close after T time units, and we might want to estimate the average delay over all customers who went to the site for service. In this case we need to introduce the counting process $\{N(t) : t \geq 0\}$ for the arrival times t_n, where $N(t)$ denotes the number of arrivals during the time interval $(0, t]$, and can be represented by $N(t) = \max\{j : t_j \leq t\}$. Thus there would be $N(T)$ arrivals during our desired time interval $(0, T]$ and we wish to estimate

$$E\Big[\frac{1}{N(T)} \sum_{j=1}^{N(T)} D_j\Big].$$

Each X_i will be a copy of

$$\frac{1}{N(T)} \sum_{j=1}^{N(T)} D_j.$$

Here $N(T)$ is a random variable so each copy X_i will typically have a different value for $N(T)$. Note that $t_{N(T)}$ is the last arrival time before (or at) time T, and $t_{N(T)+1}$ is the first arrival time strictly after time T;

$$t_{N(T)} \leq T < t_{N(T)+1}.$$

For each run (yielding a copy X_i) we would thus simulate our delays D_j until $j = N(T)$, and then stop. Using the fact that $t_{j+1} = t_j + T_j,\ j \geq 0$, we keep simulating the D_j until finally (for some j) $t_{j+1} > T$, that is we set $N = N(T) = \min\{j : t_{j+1} > T\}$ and we stop simulating this run yielding our copy X_i as

$$\frac{1}{N} \sum_{j=1}^{N} D_j.$$

Repeating this procedure n times yields our copies $X_1, \ldots, X_n$.

In general, estimating an expected value such as $\theta = E(D_5)$ or an integral such as $\theta = \int_0^1 g(x)dx = E(g(U))$ by generating a large number, n, of copies $X_1, \ldots, X_n$, with mean $E(X_i) = \theta$, and then using the empirical average in (2) as the estimate is known as *Monte Carlo* simulation. The name apparently comes historically from the fact that the generation of random numbers needed could be provided in the context of gambling, in the famous gambling city of Monte Carlo, as the outcome of a *roulette wheel*.

We will study Monte Carlo simulation in more detail later, where we will study methods that purposely introduce correlation among the copies X_j so as to reduce the variance of the estimator.