Efficiency-Improvement Techniques

Ref: Law and Kelton, Chapter 11

1. Variance Reduction and Efficiency Improvement

The techniques that we will look at typically are designed to reduce the variance of the estimator for the performance measure that we are trying to estimate via simulation. (Reduction of the variance leads to narrower confidence intervals, and hence less computational effort is required to achieve a given level of precision.) The reduction is measured relative to the variance obtained when using “straightforward simulation.” For this reason these techniques are sometimes called variance-reduction methods.

One must be careful when evaluating these techniques: they are only worthwhile if the reduction in computer effort due to the variance reduction outweighs the increase in computational effort needed to execute the technique during the simulation, as well as the additional programming complexity.

For many of the methods that we will look at, it is obvious that the additional effort to implement the variance reduction is small, so the technique is a clear win. How do we deal with more complicated situations?

One fair way to compare two estimation methods is to assume a fixed computer budget of $c$ time units, and compare the confidence-interval width for the two methods at the end of the simulation.

Example:

Suppose $\alpha = E[X] = E[Y]$. Should we generate i.i.d. replicates of $X$ or $Y$ in order to estimate $\alpha$?

Let $c$ be the computer budget. Then, the number of $X$-observations generated within budget $c$ is

$$N_X(c) = \max \{n \geq 0 : \tau_X(1) + \ldots + \tau_X(n) \leq c\},$$

where $\tau_X(i)$ is the (random) computer time required to generate $X_i$. Assume (reasonably) that $(X_1, \tau_X(1)), (X_2, \tau_X(2)), \ldots$ is a sequence of i.i.d. pairs. It can be shown that

$$\frac{1}{c}N_X(c) \to \lambda_X$$

with probability 1 as $c \to \infty$, where $\lambda_X = \frac{1}{E[\tau_X]}$. Now, note that

$$\alpha_X(c) = \frac{1}{N_X(c)} \sum_{i=1}^{N_X(c)} X_i \quad (\text{set } \alpha_X(c) = 0 \text{ if } N_X(c) = 0)$$

is the estimator for $\alpha$ (based on $X$) obtained after expending budget $c$. Since $N_X(c) \approx \lambda_X \cdot c$, it follows that
\[ \alpha_X(c) - \alpha = \frac{1}{N_X(c)} \sum_{i=1}^{N_X(c)} X_i - \alpha = \frac{1}{\lambda_X c} \sum_{i=1}^{\lambda_X c} X_i - \alpha = \mathcal{D} \frac{\text{Var}[X]}{\lambda_X c} N(0,1) \]

Similarly, if \( \alpha_Y(c) \) is the estimator based on i.i.d. replications of \( Y \),

\[ \alpha_Y(c) - \alpha = \mathcal{D} \frac{1}{\sqrt{c}} \sqrt{\mathbb{E}[\tau_X] \cdot \text{Var}[X]} N(0,1). \]

Clearly, we should choose the estimator that minimizes the product of the mean computer time per observation and the variance per observation. This conclusion holds more generally, so the inverse of this product is a reasonable measure of efficiency.

2. Common Random Numbers (CRN)

This method applies when comparing two or more alternate system configurations.

Intuition

The overall intuition is that you get a better comparison between two systems if they are both subjected to the exact same random fluctuations, since then the measured differences will be due to true differences in performance.

In more detail, suppose that under two possible system configurations, the performance measure is distributed as \( X \) and \( Y \), respectively, and we wish to estimate \( \alpha = E[X] - E[Y] = E[X - Y] \). Suppose we have simulated both systems to obtain i.i.d. replicates \( (X_1, Y_1), \ldots, (X_n, Y_n) \). Then our estimate of \( \alpha \) is \( \alpha_n = \frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i) \) and

\[ \text{Var}[\alpha_n] = \frac{1}{n} \text{Var}[X - Y] = \frac{1}{n} \left( \text{Var}[X] + \text{Var}[Y] - 2 \text{Cov}[X, Y] \right) \]

If the \( X_i \)'s are simulated independently of the \( Y_i \)'s, then \( \text{Cov}[X, Y] = 0 \) (this is the standard approach). If we could somehow arrange things so that \( \text{Cov}[X, Y] > 0 \), then the variance of \( \alpha_n \) would be reduced.

Suppose for simplicity that exactly one uniform random number is used to generate \( X_i \) and similarly for \( Y_i \). If we use the *same* random number for both \( X_i \) and \( Y_i \), then we can write \( X_i = X_i(U_i) \) and \( Y_i = Y_i(U_i) \), where each \( U_i \) is a uniform[0,1] random number. If \( X_i(u) \) and \( Y_i(u) \) are both increasing in \( u \) or both decreasing in \( u \), then \( \text{Cov}[X_i, Y_i] > 0 \), and we have variance reduction. (Caveat: in general, if CRN is not done correctly, the covariance between the simulations can be negative, and the variance of \( \alpha_n \) will *increase*.)
In Practice:

To simulate a real system using this idea, we try to synchronize the random numbers between the two systems as much as possible. This usually entails using several streams of pseudorandom U[0,1] numbers, one for each type of event in the system.

Example:

Consider the problem of comparing the long run average waiting time for two GI/G/1 queues. In both systems, the interarrival times are i.i.d. according to a distribution function $G$, and in system $i$ ($i=1, 2$) the successive service times are i.i.d. according to a distribution function $H_i$.

To apply CRN, we use two streams of pseudorandom numbers, $(U_j: j \geq 0)$ and $(V_j: j \geq 0)$. For both systems, we generate the $j^{th}$ interarrival time by using $U_j$ to generate a sample from $G$. (I.e., jobs arrive at the same time in both systems.) Although the jobs arrive at the same time, we can’t make the service times match exactly. However, suppose we use inversion and generate $T_{ij}$ the $j^{th}$ service time for system $i$ by $T_{ij} = H_i^{-1}(V_j)$. Since $H_i^{-1}(\cdot)$ and $H_2^{-1}(\cdot)$ are both increasing functions, we have $\text{Cov}[T_{1,j}, T_{2,j}] > 0$.

Note that, if we used only a single stream of random numbers, things would soon get out of sync, with the same pseudorandom number being used to generate an interarrival in one system and a service time in a different system.

3. Antithetic Variates

The idea is similar to CRN, but now we consider a single system with a performance measure $X$. Suppose that we simulate the system to get identically distributed replicates $X_1, \ldots, X_{2n}$ and estimate $E[X]$ by the usual estimator $\alpha_{2n} = \overline{X}_{2n}$. Also suppose that the $n$ pairs $(X_1, X_2), (X_3, X_4), \ldots, (X_{2n-1}, X_{2n})$ are independent, but the elements of each pair might be correlated. We have

$$\text{Var}[\alpha_{2n}] = \frac{1}{4n^2} \left( \text{Var}[X_1] + \cdots + \text{Var}[X_{2n}] + 2\text{Cov}[X_1, X_2] + \cdots + 2\text{Cov}[X_{2n-1}, X_{2n}] \right)$$

If all of the simulations are independent, then the covariance terms are all equal to 0. If we can arrange the simulations so that $\text{Cov}[X_{2k-1}, X_{2k}] < 0$ for $1 \leq k \leq n$, then we will have variance reduction. This is the “opposite” of CRN.

Intuition: For every simulation with “favorable” random fluctuations, there is a matching simulation with “unfavorable” conditions, so we don’t just see one kind of performance.

The following example shows two ways in which antithetic variates can be implemented.

Example (GI/G/1 Queue):
Consider a GI/G/1 queue as in the CRN example (with interarrival time distribution G and service-time distribution H), and suppose as before that we wish to estimate the average waiting time of the first 100 customers. Denote by $X_k$ the average waiting time of the first 100 customers during the $k$th simulation replication. To generate $(X_{2k-1}, X_{2k})$ (where $1 \leq k \leq n$), use a single stream $(U_{j,k} : j \geq 1)$ to generate interarrival times for both the $(2k-1)^{st}$ and $k^{th}$ simulation replications and similarly use a single stream $(V_{j,k} : j \geq 1)$ to generate service times. Now, however, generate the $j^{th}$ interarrival time and the $j^{th}$ service time for the $(2k-1)^{st}$ simulation replication as $G^{-1}(U_{j,k})$ and $H^{-1}(V_{j,k})$, respectively, and the $j^{th}$ interarrival time and the $j^{th}$ service time for the $(2k)^{th}$ simulation replication as $G^{-1}(1-U_{j,k})$ and $H^{-1}(1-V_{j,k})$, respectively. This will induce the desired negative correlation.

The above idea works in a wide variety of contexts. For this specific queueing model, another way of inducing negative correlation is to let the $j^{th}$ service time for replication $2k – 1$ be the $j^{th}$ interarrival time for replication $2k$. (This works because a long interarrival time decreases delays while a long service time increases delays.) In generally, some sort of monotonic response is required for antithetic random numbers to work.

**Warning:** Bratley, Fox, and Schrage, p. 57 point out that combining CRN and antithetic variates can produce bad results if not done carefully.

### 4. Control Variates

Suppose that we wish to estimate $E[X]$, and there exists a random variable $Y$ such that $E[Y]$ can be computed analytically. Define a “control variable” $C = Y – E[Y]$ and consider the “controlled estimator”

$$X(\lambda) = X - \lambda C,$$

Since $E[C] = 0$, we have $E[X(\lambda)] = \alpha$ for all $\lambda$. Hence, we may choose $\lambda$ so as to minimize the variance of $X(\lambda)$. Set

$$v(\lambda) = \text{var}[X(\lambda)] = \text{var}[X] - 2\lambda \text{Cov}[X, C] + \lambda^2 \text{Var}[C].$$

This is a quadratic function in $\lambda$ with a minimum at

$$\lambda^* = \frac{\text{Cov}[X, C]}{\text{Var}[C]} = \frac{E[(X - \mu_X)C]}{E[C^2]} \quad (\text{since } E[C] = 0).$$

The minimizing variance is then

$$v(\lambda^*) = (1 - \rho^2) \text{Var}[X].$$
where $\rho$ is the coefficient of correlation between $X$ and $C$, i.e. $\rho = \frac{\text{Cov}[X, C]}{\sqrt{\text{Var}[X] \cdot \text{Var}[C]}}$. This establishes that a significant variance reduction is possible if $\rho$ is close to either +1 or -1 or, equivalently, if $X$ and $C$ have a strong linear relationship.

Of course, $\lambda^*$ is unknown, but we can estimate it by

$$\lambda^*_n = \frac{1}{n-1} \frac{\sum_{i=1}^{n} (X_i - \bar{X}_n)C_i}{\sum_{i=1}^{n} C_i^2},$$

where $(X_1, C_1), \ldots, (X_n, C_n)$ are i.i.d. replicates of $(X, C)$.

Finally, we apply our usual estimation methods for i.i.d. random variables to the sequence $Z_1, \ldots, Z_n$, where $Z_i = X_i - \lambda^*_n C_i$ for $1 \leq i \leq n$.

**Example:**

For the GI/G/1 queue, suppose we wish to estimate the expected average delay for the first $n$ customers. Then for the $i^{th}$ simulation, $X_i$ is the average delay for the first $n$ customers. Denote by $V_{k,i}$ the $k^{th}$ service time during the $i^{th}$ simulation, and suppose the $\mathbb{E}[V_{k,i}] = 5$. Then we can take

$$C_i = \frac{1}{n} \sum_{k=1}^{n} V_{k,i} - 5$$

to be the control variable for the $i^{th}$ simulation.

**Multiple Controls**

The above idea can be extended to the case of multiple control variates $C^{(1)}, \ldots, C^{(m)}$: the controlled estimator is of the form

$$X(\lambda_1, \ldots, \lambda_m) = X - \lambda_1 C^{(1)} - \ldots - \lambda_m C^{(m)}.$$

The formulas for $(\lambda^*_1, \ldots, \lambda^*_m)$ are now found by solving a system of $m$ linear equations; see Law and Kelton, p. 609. (We are essentially fitting a regression model, and simulating for the “leftover” uncertainty.)
5. Importance Sampling

Suppose that we wish to estimate \( \alpha = E[g_n(X_0, X_1, ..., X_n)] \), where \( X_0, X_1, ..., X_n \) are i.i.d. replicates of a discrete random variable \( X \) with state space \( S \). As discussed previously, if \( Y \) is another random variable with state space \( S \) that is “easier” to simulate than \( X \), then we can generate i.i.d. replicates \( Y_0, ..., Y_n \) of \( Y \) and use the fact that \( E[g_n(Y_0, Y_1, ..., Y_n) L_n] = E[g_n(X_0, X_1, ..., X_n)] \), where

\[
L_n = \frac{\prod_{i=0}^{n} p(Y_i)}{\prod_{i=0}^{n} q(Y_i)}.
\]

is the likelihood ratio.

Note that if \( g_n(Y_i, ..., Y_n) = \prod_{i=0}^{n} g(Y_i) \) for some function \( g \) and if we choose \( q(s) = g(s)p(s)/\alpha \) for \( s \in S \), then \( g_n(Y_0, Y_1, ..., Y_n)L_n = \alpha \), so that our modified estimator has variance 0. Of course, in practice we cannot achieve such “perfect” variance reduction since we don’t know \( \alpha \), but the foregoing observation sometimes can provide guidance when searching for the best importance-sampling distribution.

Application to Efficiency Improvement: Rare Events

Consider a DTMC model of the state of a machine. The state space is \( S = \{0, 1, 2, 3\} \). \( X_n = 0 \) if the machine is running and fully operational at time \( n \) and \( X_n = 3 \) if the machine has failed at time \( n \). States 1 and 2 correspond to states in which the machine is running but performance has degraded, with 2 being a worse state than 1. Suppose that the initial distribution \( \mu \) is given by \( \mu(0) = 1 \) and the state-transition probabilities are given by

\[
\begin{bmatrix}
0 & 1 & 2 & 3 \\
0 & \mu & 0 & 0 \\
\frac{\lambda}{\lambda + \mu} & 0 & \frac{\lambda}{\lambda + \mu} & 0 \\
0 & \frac{\mu}{\lambda + \mu} & 0 & \frac{\lambda}{\lambda + \mu} \\
0 & 0 & 1 & 0 
\end{bmatrix}
\]

and that \( \mu >> \lambda \), so that failures take a long time to occur.

Suppose that we wish to estimate \( \alpha = P\{N \leq j\} = E[I(N \leq j)] \), where \( N = \min\{n > 0: X_n = 3\} \) is the time to the first failure and \( j \) is a small to medium-valued integer. Because failures take a long time to occur, the event \( A = \{N \leq j\} \) occurs very infrequently, so it is hard to estimate the probability of \( A \). Note that we can write \( \alpha = E[g(X_0, ..., X_j)] \), where
g_i(x_0, \ldots, x_j) = \begin{cases} 1 & \text{if } x_i = 3 \text{ for some } 0 \leq i \leq j \\ 0 & \text{otherwise.} \end{cases}

Importance sampling works very well for rare-event simulations. The idea is to simulate using a transition matrix $\tilde{P}$ under which early failures happen relatively frequently. It turns out that a good choice is to make $\mu$ equal to $\lambda$, so that a “repair type” transition is as likely as a “failure type” transition.

So we take

$$\tilde{P} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0.5 & 0 & 0.5 & 0 \\ 2 & 0 & 0.5 & 0 & 0.5 \\ 3 & 0 & 0 & 1 & 0 \end{pmatrix}. \quad (4)$$

The initial distribution is $\nu = \mu$.

**Algorithm**

1. Choose a sample size $n$.
2. Using initial distribution $\nu$ and transition matrix $\tilde{P}$ as specified by (4), simulate the DTMC $\{Y_n: n \geq 0\}$ up until min(time $j$, a failure occurs); denote the time to failure by $N$.
3. Compute $W = I(N \leq j) \left( \prod_{i=1}^{N} P(Y_{i-1}, Y_i) \right) \left/ \prod_{i=1}^{N} \tilde{P}(Y_{i-1}, Y_i) \right.$
4. Repeat steps 2 and 3 $n$ independent times, thereby yielding i.i.d. replicates $W_1, \ldots, W_n$
5. Compute point estimates and confidence intervals as usual.

**Remark:** The importance-sampling technique also can be applied to GSMP’s by initially setting the likelihood ratio $L = 1$ and updating $L$ as the simulation proceeds. For example, if a new clock reading $C_{n,i}$ (in the modified GSMP) is generated for event $e_i$ at the $n^{th}$ state-transition time, then $L$ gets multiplied by $f(C_{n,i}; S_n, e_i, S_{n-1}, E_n^*) / \tilde{f}(C_{n,i}; S_n, e_i, S_{n-1}, E_n^*)$, where $f$ and $\tilde{f}$ are the densities of the clock-setting distribution functions in the original and modified GSMP’s, respectively, and $E_n^*$ is the set of transitions that trigger the $n^{th}$ state transition. Similarly, whenever a new state $S_n$ is generated, $L$ is multiplied by $p(S_n; S_{n-1}, E_n^*) / \tilde{p}(S_n; S_{n-1}, E_n^*)$.

6. **Conditional Monte Carlo**
Example

Let \((X(t): t \geq 0)\) be a CTMC with state space \(S\) and rate matrix \(Q\), and suppose that we wish to estimate \(\alpha = E[Z]\), where

\[
Z = \int_{0}^{T_n} f(X(u)) \, du ,
\]

and \(T_n\) is the time of the \(n^{th}\) state transition of the process.

The usual way to generate an observation of \(Z\) is to in effect generate the successive states of the chain \(X_0, X_1, \ldots, X_{n-1}\) (via a DTMC with transition matrix \(R\) as in Lecture Notes #2) and then generate the holding times; the \(i^{th}\) holding time \((i \geq 0)\) is exponentially distributed with mean \(1/q(X_i)\) where, for \(x \in S\), \(q(x) = -Q(x, x)\) is the rate at which the CTMC jumps out of state \(x\).

A key observation is that we can reduce the variance by replacing the randomly generated holding times with their expected values.

This type of variance-reducing trick is known as Conditional Monte Carlo. It hinges on the general facts that for a random variable \(U\) and a random vector \(V\)

\[
E[E[U | V]] = E[U] \quad \text{and} \quad \text{Var}[U] = \text{Var}[E[U | V]] + E[\text{Var}[U | V]] \geq \text{Var}[E[U | V]].
\]

(Exercise: prove the second identity.) Thus, if we can simulate \(V\) and compute \(\tilde{U} = E[U | V]\), then \(\tilde{U}\) has the same mean as \(U\), but lower variance, so we should obtain i.i.d. replicates of \(\tilde{U}\) to estimate \(\alpha = E[U]\) rather than i.i.d. replicates of \(U\).

For our particular example, we have \(U = Z\) and \(V = (X_0, \ldots, X_n)\). Thus, we want to estimate \(E[\tilde{Z}] (=E[Z])\), where

\[
\tilde{Z} = E[Z | X_0, \ldots, X_n] = \sum_{i=0}^{n-1} f(X_i) \frac{1}{q(X_i)}.
\]

We can simulate the DTMC \((X_n: n \geq 0)\) \(n\) independent times to obtain i.i.d. replicates \(\tilde{Z}_1, \ldots, \tilde{Z}_k\). Point estimates and confidence intervals for \(E[\tilde{Z}]\) can then be obtained in the usual way.