Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

COMPUTATIONAL EFFICIENCY EVALUATION IN OUTPUT ANALYSIS

Halim Damerdji
Department of Industrial Engineering
North Carolina State University
Raleigh, North Carolina 27695–7906, U.S.A.

Shane G. Henderson
Department of Industrial and Operations Engineering
University of Michigan
Ann Arbor, Michigan 48109–2117, U.S.A.

Peter W. Glynn
Department of Engineering-Economic Systems and Operations Research
Stanford University
Stanford, California 94305–4023, U.S.A.

ABSTRACT

A central quantity in steady-state simulation is the time-average variance constant. Estimates of this quantity are needed (for example) for constructing confidence intervals, and several estimators have been proposed, including nonoverlapping and overlapping batch means methods, spectral methods, and the regenerative method. The asymptotic statistical properties of these estimators have been investigated but the computational complexity involved in computing them has received very little attention.

We assume a fixed simulation run-length, as opposed to sequential methods in which the run-length is determined dynamically. In order to consistently estimate the time-average variance constant, all of the estimators require an amount of computation that is linear in the time-horizon simulated, with the exception of spectral methods which require a superlinear amount of computation.

1 INTRODUCTION

Let $X = (X_n : n \geq 1)$ be a real-valued stochastic process evolving in discrete time. Under very general conditions it is known that

$$\alpha(n) \stackrel{\triangle}{=} \frac{1}{n}\sum_{i=1}^{n} X_i \to \alpha \quad \text{a.s.}$$

as $n \to \infty$, where α is a deterministic constant. We refer to the problem of estimating α and obtaining related error bounds through simulation as the *steady-state estimation problem*. Again, under very general conditions, $\alpha(n)$ satisfies a central limit theorem (CLT) of the form

$$\sqrt{n}(\alpha(n) - \alpha) \stackrel{\mathcal{D}}{\to} \sigma N(0,1), \tag{1}$$

where $\stackrel{\mathcal{D}}{\to}$ denotes convergence in distribution, $N(0,1)$ is a standard normal random variable (r.v.), and σ^2 is the *time-average variance constant* (TAVC) of the process. When the process is covariance stationary, the TAVC is essentially the sum of the covariances at all lags; see Section 5 for the definition of covariance stationarity.

Equation (1) provides a basis for computing a confidence interval for α based on $\alpha(n)$ if an estimator for σ^2 can be computed. Several estimators for σ^2 have been proposed in the literature; see chapter 3 of Bratley, Fox and Schrage (1987), or chapter 8 of Law and Kelton (1991) for an overview. In selecting an estimator $V(n)$ for σ^2, one typically takes into account the properties of X (e.g., is it easy to define regeneration epochs for X?). Of course, one should also take into account the computational effort involved in computing $V(n)$, as the following analysis shows (see also Glynn and Whitt, 1992).

Suppose that one has a computational budget of c units of computer time, and that each transition (each unit of simulated time) costs a fixed s units of computer time to simulate. Suppose further that for a unit increase in simulated time, the computational effort required to compute $V(n)$ increases by v, so that computing $V(n)$ requires vn units of computer time. (The assumption that both the simulation effort and the computational effort in computing $V(n)$ increase exactly linearly in n may seem somewhat restrictive, but the following analysis is easily generalized to the case where, for example, the average

effort over the first n transitions converges weakly to a constant.) Let $\bar{\alpha}(c)$ be the estimator of α obtained from c units of computer time, and note that if we include the time to compute $V(n)$, $\lfloor c/(s+v) \rfloor$ transitions of X will have been recorded. Therefore, $\bar{\alpha}(c) = \alpha(\lfloor c/(s+v) \rfloor)$. Now note that if the CLT (1) holds, then

$$\begin{aligned} \sqrt{c}(\bar{\alpha}(c) - \alpha) &= \sqrt{c}\{\alpha(\left\lfloor \frac{c}{s+v} \right\rfloor) - \alpha\} \\ &\xrightarrow{\mathcal{D}} \sqrt{s+v}\ \sigma N(0,1) \end{aligned}$$

as $c \to \infty$. We can now see the effect of the effort required to compute $V(n)$. A confidence interval for α based on this CLT will have a half-width that is proportional to $\sqrt{s+v}$, so that the effort required to compute the TAVC estimator (quantified by v) has a direct impact on the accuracy of the point estimator $\bar{\alpha}(c)$ available after c units of computer budget have been expended. If we could reduce v, we could also reduce the statistical error in the estimator $\bar{\alpha}(c)$. (A similar analysis can be performed in the case where the effort to compute $V(n)$ increases at a super-linear rate.)

Clearly then, one should be aware of the computational requirements of various TAVC estimators. In this paper we evaluate the computational burden involved in computing TAVC estimators. We focus on fixed run-length procedures where the simulation run-length is decided in advance. We will address sequential methods (where the simulation is run until some stopping criteria are met) in a future paper.

Algorithms for computing a TAVC estimator may be grouped into two categories: single pass and multipass. In multipass algorithms, two or more passes through the data are required to compute the TAVC. In single pass algorithms only one sweep is made through the data. One computes $V(n+1)$ (or $V(n+q)$ for some integer q) in terms of $V(n)$. Single pass algorithms require relatively little storage and also have the advantage that they may be efficiently applied in sequential methods (that we do not consider here). It may also be the case that a single pass algorithm is computationally more efficient than a multipass method. Therefore, although we do not discuss sequential methods in this paper, we do explore one pass algorithms.

Below, we will not distinguish between the computational effort involved in performing different arithmetic operations, i.e., between performing an addition, a subtraction, a multiplication, and a division. An arithmetic operation of this form will be denoted by `flop` and storage of a number by `stor`.

Cancellation methods may also be used to compute confidence intervals, but they do not estimate the TAVC, so we do not consider them here. The work in this paper is devoted to discrete-time processes. In Section 2 we evaluate several basic algorithms for computing sample means and variances. The nonoverlapping batch means method appears in Section 3, and overlapping batch means are considered in Section 4. The spectral method is evaluated in Section 5, and the regenerative method appears in Section 6.

2. TERMINATING SIMULATIONS

In this section we introduce algorithms for computing sample means and variances for terminating simulations, and describe their computational properties. These algorithms are not out of place in this paper, since they are often used as subroutines for computing TAVC estimators.

First, we consider how to compute the average of a large number of observations. Call $L_1,\ldots,L_q$ these observations and $\bar{L}_q$ their sample average, i.e.,

$$\bar{L}_q = \frac{1}{q}\sum_{i=1}^{q} L_i. \tag{2}$$

Borrowing notation from Chan, Golub, and LeVeque (1983), let, for $1 \le i \le j \le q$,

$$T_{i,j} = \sum_{r=i}^{j} L_r,$$

$$M_{i,j} = \frac{1}{j-i+1} T_{i,j},$$

and

$$S_{i,j} = \sum_{r=i}^{j} (L_r - M_{i,j})^2.$$

The most natural way to compute (2) is

Procedure sum.1:
$T_{1,1} = L_1$
for $i = 2$ to q, do
  $T_{1,i} = T_{1,i-1} + L_i$
return $T_{1,q}/q$

whose computational effort is q `flop`'s. The following alternative procedure is more stable numerically. It is more taxing computationally though, as it requires $3q$ `flop`'s.

Procedure sum.2:
$M_{1,1} = L_1$
for $i = 2$ to q, do
  $y = (L_i - M_{1,i-1})/i$
  $M_{1,i} = M_{1,i-1} + y$
return $M_{1,q}$

Example: One hundred million independent and identically distributed (i.i.d.) r.v.'s with a uniform(0,1000) distribution were generated. By the strong law of large numbers, the sample average should be close to the theoretical mean 500. Using single-precision floating point arithmetic, serious numerical errors occurred under Procedure sum.1, which produced a sample average of (171.798). The sample average was (499.939) under Procedure sum.2. However, under double-precision floating-point arithmetic, the two procedures led to numbers equal up to nine decimal places.

This example illustrates the need to use double-precision arithmetic whenever dealing with a large number of real-valued numbers. We will henceforth assume that all averages are computed using procedure sum.1 and double precision arithmetic unless explicitly stated otherwise. For a discussion of modern summation methods, see Higham (1993).

A number of algorithms have appeared for efficient computation of sample variances. We will focus on computation of

$$S_{1,q} = \sum_{i=1}^{q}(L_i - \bar{L}_q)^2 \quad (3)$$

$$= \sum_{i=1}^{q} L_i^2 - (1/q)\Big(\sum_{i=1}^{q} L_i\Big)^2. \quad (4)$$

The procedure given next is based on (3) and so requires passing twice through the series of observations.

Procedure var.1:

compute $\bar{L}_q$ (first pass)
sum = 0.0
for $i = 1$ to q, do
  sum=sum+$(L_i - \bar{L}_q)^2$
return sum

This two-pass algorithm involves having to store the q numbers $L_1,\ldots,L_q$ in core memory, if possible, or on secondary storage. The computing effort is $4q$ `flop`'s and q `stor`'s. The procedure is very stable numerically but does require the storage of a large number of observations. Another disadvantage of the two-pass method is that it is impractical in a sequential sampling setting when one needs to update the sample variance every time an additional observation is sampled (or every so often).

The most computationally efficient way to compute $S_{1,q}$ is via (4); the so-called textbook one-pass algorithm requires $3q$ `flop`'s and no storage. It is not a recommended procedure, however, especially when the L_i's are close to one another (i.e., when their variance is small). The following procedure, due to Hanson (1975), is one of the recommended ones in the literature, and is based on the induction

$$S_{1,i} = S_{1,i-1} + \frac{(i-1)}{i}(L_i - \bar{L}_{i-1})^2, \quad (5)$$

with $S_{1,1}$=0.

Procedure var.2:

$M_{1,1} = L_1$, $S = 0.0$
for $i = 2$ to q, do
  $y = L_i - M_{1,i-1}$
  $z = y/i$
  $M_{1,i} = M_{1,i-1} + z$
  $S = S + (i-1)yz$
return S ($S_{1,q} = S$)

The procedure requires $6q$ `flop`'s and no storage. This single-pass method is stable numerically, and is very useful in a sequential setting. Another efficient and stable procedure for computing the sample variance is the pairwise algorithm, developed by Chan, Golub, and LeVeque (1982). Other relevant references include Chan and Lewis (1979), Clark (1980), West (1979), and Youngs and Cramer (1971).

3. NONOVERLAPPING BATCH MEANS

By fixed sample size method, it is meant that before the start of the simulation, the analyst decides the total number of process observations to be simulated. (Naturally, the analyst has no control afterwards on the width of the confidence interval.) The n process observations are divided up into k adjacent batches, each of size m, assuming $km = n$ for simplicity. For $a \geq 0$, $b \geq 1$ and $a + b \leq n$, consider

$$\overline{Y}_{a,b} \triangleq \frac{1}{b}\sum_{i=a+1}^{a+b} X_i.$$

The jth batch mean (j=1,...,k) is $\overline{Y}_{(j-1)m,m}$ and the grand sample mean $\overline{Y}_{0,n}$. The batch means TAVC estimator is

$$V_{\rm bm}(n) = \frac{m}{k-1}\sum_{j=1}^{k}\Big(\overline{Y}_{(j-1)m,m} - \overline{Y}_{0,n}\Big)^2 \quad (6)$$

$$= \frac{n}{k-1}\Big(\frac{1}{k}\sum_{j=1}^{k}\overline{Y}^2_{(j-1)m,m} - \overline{Y}^2_{0,n}\Big). \quad (7)$$

It is known that when the batch size and number of batches get large (at a suitable rate) with the sample size, the batch means variance estimator is consistent

(under certain conditions on the process). See Carlstein (1986) and Damerdji (1994).

It is straightforward to evaluate the computational efficiency for this estimator. Using a two-pass algorithm, the first pass to compute the batch means and grand sample mean, and the second pass to compute (6), it is necessary to store the k values of $\overline{Y}_{(j-1)m,m}$ for j=1,...,k. The computational effort will be $n+4k$ `flop`'s and k `stor`'s. Assuming an infinite precision, one would use the textbook formula (7) which requires $n+3k$ `flop`'s and no storage. Because the $\overline{Y}_{(j-1)m,m}$'s are very close to one another for a large batch size, the procedure is unstable numerically. A numerically stable single-pass procedure would use Procedure var.2 applied to the batch means. The computational effort is $n+6k$ `flop`'s with no storage.

If one wishes to minimize the mean squared error of the TAVC estimator $V_{\rm bm}(n)$, Goldsman and Meketon (1986) showed that the optimal batch size $m = cn^{1/3}$ for a certain process constant c. Furthermore, Glynn and Whitt (1991) showed that to obtain a consistent TAVC estimator using batch means, a necessary condition is that the number of batches $m \to \infty$ as $n \to \infty$. In either case we require that m grow without bound as $n \to \infty$. Under this requirement, the dominant computational effort needed to compute $V_{\rm bm}(n)$ is still $O(n)$ (i.e., the number of `flop`'s is bounded above by a linear function of n).

We have been assuming that once the sample size n is fixed, the analyst decides upon a batch size and then computes the corresponding TAVC estimator. If the analyst desires to try different batch sizes with that same total number of observations, it then becomes necessary to store all the observations or a number of intermediate batch means with a carefully selected sampling plan. A convenient choice is to consider a total number of observations and batch size that are powers of two; one could then easily consider a new batch size that is a power (positive or negative) of two of the old batch size.

4- OVERLAPPING BATCH MEANS

This method of steady-state output analysis is a consistent-estimation method, introduced in the simulation context by Meketon and Schmeiser (1984). The OBM variance estimator is given by

$$V_{\rm obm}(n) = \frac{m}{n-m+1}\sum_{j=0}^{n-m}\Big(\overline{Y}_{j,m}-\overline{Y}_{0,n}\Big)^2 \quad (8)$$

$$\approx \frac{m}{n-m+1}\Big(\sum_{j=0}^{n-m}\overline{Y}_{j,m}^2 - n\overline{Y}_{0,n}^2\Big). \quad (9)$$

The textbook formula (9) requires storage of m successive observations and about $4n$ `flop`'s. The two-pass algorithm of (8) requires on the order of $6n$ `flop`'s and necessitates storing the n observations in either core memory or on a storage device (the space can be reused to store the $n-m+1$ centered batch means). An alternative to having to store all of the observations is to use the following single-pass procedure which does require, however, storage of the last m observations (but, then, so does the textbook formula). For $r \geq m+1$, let

$$\Delta(r) = \sum_{j=0}^{r-m}\Big(\overline{Y}_{j,m}-\overline{Y}_{0,r}\Big)^2 = \frac{r-m+1}{m}V_{\rm obm}(r),$$

$$\Lambda(r) = \sum_{j=0}^{r-m}\Big(\overline{Y}_{j,m}-\overline{Y}_{0,r}\Big),$$

and

$$\Psi(r) = \overline{Y}_{0,r+1}-\overline{Y}_{0,r} = \Big(X_{r+1}-\overline{Y}_{0,r}\Big)/(r+1).$$

We have the relations

$$\begin{aligned}\Delta(r+1) &= \Delta(r)+(r-m+2)\Psi^2(r)-2\Psi(r)\Lambda(r)\\ &\quad -2\Psi(r)\Big(\overline{Y}_{r-m+1,m}-\overline{Y}_{0,r}\Big)\\ &\quad +\Big(\overline{Y}_{r-m+1,m}-\overline{Y}_{0,r}\Big)^2, \qquad (10)\end{aligned}$$

and

$$\begin{aligned}\Lambda(r+1) &= \Lambda(r)-(r-m+2)\Psi(r)\\ &\quad +\Big(\overline{Y}_{r-m+1,m}-\overline{Y}_{0,r}\Big). \qquad (11)\end{aligned}$$

One should first compute $\Delta(m+1)$ and $\Lambda(m+1)$. After sampling X_{r+1} ($r \geq m+1$), one uses (10) and (11) to update the variables. It takes 3 `flop`'s to compute $\Psi(r)$ and an additional `flop` to compute $\overline{Y}_{0,r+1}$ (using procedure sum.2). It takes 4 `flop`'s to compute $\overline{Y}_{r-m+1,m}-\overline{Y}_{0,r}$, 9 `flop`'s for $\Delta(r+1)$, and 4 `flop`'s for $\Lambda(r+1)$. At each iteration, 21 `flop`'s are performed. The total effort to compute $V_{\rm obm}(n)$ is about $21n$ `flop`'s and m `stor`'s. The two major advantages of this iterative procedure are that (1) one does not have to store all the observations, and (2) it is very efficient in a sequential sampling setting. This procedure is computationally more demanding, but its major drawback is that the batch size cannot be modified in a sequential setting. If the analyst wants to try different batch sizes for a fixed number n of observations, then all these n observations must be stored.

For the overlapping batch means estimator to be a consistent estimator of the TAVC, the batch size

m must be such that $m \to \infty$ and $m/n \to 0$ as $n \to \infty$ (among other conditions) (Damerdji 1994). Goldsman and Meketon (1986) (see also Song and Schmeiser, 1995) show that to minimize the mean squared error of $V_{\rm obm}(n)$, the batch size $m = m(n) = cn^{1/3}$ for a certain process constant c. The number of flops required to compute $V_{\rm obm}(n)$ remains $O(n)$ when $m = cn^{1/3}$.

5. THE SPECTRAL METHOD

5.1. The Spectral Variance Estimator

Let us assume that the process is covariance stationary, i.e., that $EX_i = \mu$ and $E[(X_i - \mu)(X_{i+q} - \mu)] \equiv \gamma(q)$ for all lags q and all times i. If $\sum_{q=-\infty}^{\infty} |\gamma(q)| < \infty$, then (Anderson 1971) σ^2 can typically be re-expressed as $\sum_{q=-\infty}^{\infty} \gamma(q)$. The spectral density function $f(\cdot)$ of the process is the Fourier transform of the covariance sequence, i.e., for $\lambda \in [-\pi, \pi]$,

$$f(\lambda) = \frac{1}{2\pi} \sum_{q=-\infty}^{\infty} \gamma(q) \cos(\lambda q).$$

At frequency 0, $2\pi f(0) = \sum_{q=-\infty}^{\infty} \gamma(q)$, and so an estimate of the spectral density function at frequency 0 provides an estimate of the TAVC σ^2. From Anderson (1971), for example,

$$V_{\rm s}(n) = \sum_{q=-(m-1)}^{m-1} w_n(q)\gamma_n(q), \qquad (12)$$

is a consistent estimator of σ^2 in the mean-square sense (under certain additional conditions), where:

$$\gamma_n(q) = \frac{1}{n} \sum_{r=1}^{n-q} (X_r - \overline{Y}_{0,n})(X_{r+q} - \overline{Y}_{0,n}), \qquad (13)$$

is the sample covariance at lag q; the weight function $w_n(\cdot)$ is even, $|w_n(\cdot)| \le 1$, with $w_n(0) = 1$; and m is a parameter, which we will call the batch size, such that $m \to \infty$ and $m/n \to 0$ as $n \to \infty$. The weight function $w_n(\cdot)$ is also called the lag-window function. We will restrict attention to lag-window functions such that $w_n(q) = 0$ for $|q| \ge m$. Examples include the modified Bartlett window and (one of) the Parzen window; these are $w_n(q) = 1 - |q|/m$ and $w_n(q) = 1 - q^2/m^2$, respectively, for $|q| \le m - 1$ and $w_n(q) = 0$ for $|q| \ge m$. The critical choice in the application of spectral methods is the batch size.

We rewrite $V_{\rm s}(n) = \gamma_n(0) + \sum_{q=1}^{m-1} 2w_n(q)\gamma_n(q)$. One could compute the spectral variance estimator by first computing $\overline{Y}_{0,n}$, subtracting it from every observation, then computing $\gamma_n(0),\ldots,\gamma_n(m-1)$ from (13), and finally, computing $V_{\rm s}(n)$ from (12). This takes $2(m+1)n$ `flop`'s and n `stor`'s, assuming it takes no effort to compute the $2w_n(q)$'s. The procedure is of course two-pass. A computationally more attractive way to compute the sample covariances is via the fast Fourier transform (FFT) algorithm.

5.2. The Partial-Sum Variance Estimator

An alternative way for computing/approximating a spectral variance estimator is the following. It is known that the OBM variance estimator is equal, but for some end-effect terms, to a spectral variance estimator with a modified Bartlett lag-window function. One can go the other way, i.e., given a lag-window kernel of the kind considered here, one can construct a generalized OBM-type variance estimator that is equal to the spectral variance estimator but for some end-effect terms. This estimator, introduced in Damerdji (1991), is called the partial-sum variance estimator because each observation, from 1 up to $(n - m + 1)$, starts a batch of size 1, a batch of size 2,..., and a batch of size m. Some notation is needed. Let

$$\begin{aligned}\Delta^2 w(k) &= w(k-1)\cos((k-1)\lambda) - 2w(k)\cos(k\lambda) \\ &\quad + w(k+1)\cos((k+1)\lambda),\end{aligned}$$

and $\alpha(k, \lambda) = k^2 \Delta^2 w(k)$. From Damerdji (1991),

$$n^{-1} \sum_{j=0}^{n-1} \sum_{k=1}^{m} \alpha(k,\lambda) \Big(\overline{Y}_{j,k} - \overline{Y}_{0,n}\Big)^2 \approx \sum_{q=-(m-1)}^{(m-1)} w(q)\gamma_n(q)\cos(\lambda q).$$

At frequency 0, define the partial-sum variance estimator as

$$V_{\rm ps}(n) = n^{-1} \sum_{j=0}^{n-1} \sum_{k=1}^{m} \alpha(k) \Big(\overline{Y}_{j,k} - \overline{Y}_{0,n}\Big)^2,$$

where $\alpha(k) \stackrel{\triangle}{=} \alpha(k, 0)$. It follows that $V_{\rm ps}(n) \approx V_{\rm s}(n)$. For the modified Bartlett window, $\alpha(m) = m$ and $\alpha(k) = 0$ for $k < m$, and so, for this lag-window function the partial-sum variance estimator is indeed the OBM variance estimator (with an asymptotically equivalent denominator).

Most of the lag-window functions in the literature depend explicitly upon m and not n. See Chapter 6 of Priestley (1981). If one initially chooses a large enough parameter m, then the following one-pass procedure could be used to compute $V_{\rm ps}(n)$. For

$r \geq m+1$, let

$$\Upsilon(r) = \sum_{j=0}^{r-m} \sum_{k=1}^{m} \alpha(k) \Big(\overline{Y}_{j,k} - \overline{Y}_{0,r} \Big).$$

We have that

$$\begin{aligned}(r+1)V_{\rm ps}(r+1) &= rV_{\rm ps}(r) - 2\Psi(r)\Upsilon(r) \\ &+(r-m+2)\Psi^2(r)\sum_{k=1}^{m}\alpha(k) \\ &+\sum_{k=1}^{m}\alpha(k)\Big(\overline{Y}_{r-m+1,k} - \overline{Y}_{0,r}\Big)^2 \\ &-2\Psi(r)\sum_{k=1}^{m}\alpha(k)\Big(\overline{Y}_{r-m+1,k} - \overline{Y}_{0,r}\Big)\end{aligned}$$

and

$$\begin{aligned}\Upsilon(r+1) &= \Upsilon(r) - (r-m+2)\Psi(r)\sum_{k=1}^{m}\alpha(k) \\ &+\sum_{k=1}^{m}\alpha(k)\Big(\overline{Y}_{r-m+1,k} - \overline{Y}_{0,r}\Big).\end{aligned}$$

To compute $V_{\rm ps}(n)$, one can initially compute $V_{\rm ps}(m+1)$ and $\Upsilon(m+1)$. One can then evaluate $V_{\rm ps}(m+2)$ and $\Upsilon(m+2)$ using the above relations, etc. The terms that are computationally taxing are

$$\sum_{k=1}^{m}\alpha(k)\Big(\overline{Y}_{r-m+1,k} - \overline{Y}_{0,r}\Big)^2$$

and

$$\sum_{k=1}^{m}\alpha(k)\Big(\overline{Y}_{r-m+1,k} - \overline{Y}_{0,r}\Big).$$

To compute these two terms, an array containing the last m observations should be kept (and updated). This procedure is single pass and requires about $6mn$ `flop`'s.

Note that both $V_{\rm s}(n)$ and $V_{\rm ps}(n)$ require on the order of mn `flop`'s to compute. As $n \to \infty$, m should also increase without bound to obtain a consistent estimator, so that the computation of these estimators requires superlinear effort in the simulation run-length. Therefore, the computational effort required to compute these estimators asymptotically dominates (for example) the nonoverlapping batch means estimator.

5.3 The Fast Fourier Transform

For simplicity, we now assume that the sample size is a power of two. One typically computes an estimate of the spectral density function using the FFT algorithm. From the theory of spectral analysis of time series,

$$\int_{-\pi}^{\pi} I_n(\theta) W_n(\lambda - \theta) d\theta \tag{14}$$

is a consistent estimator (in the mean-square sense) of the spectral density function at frequency λ, where $I_n(\lambda)$ is the sample periodogram and $W_n(\lambda)$ is the spectral window function (associated with some lag window function) at frequency λ. For example, the spectral window function associated with the modified Bartlett lag window function is

$$W_n(\lambda) = (1/(2\pi m))(\sin(m\lambda/2)/\sin(\lambda/2))^2.$$

More notation is needed. Consider the finite Fourier transform

$$d(\lambda) = \frac{1}{\sqrt{2\pi n}} \sum_{r=1}^{n} \Big(X_r - \overline{Y}_{0,n} \Big) \exp(-i\lambda r).$$

We have that

$$I_n(\lambda) = |d(\lambda)|^2. \tag{15}$$

Consider the frequencies $\lambda_p = 2\pi p/n$ for $p = 0, \pm 1, \ldots, \pm n/2$. The integral of (14) can then be approximated by $(2\pi/n)\sum_{p=-n/2}^{n/2} I_n(\lambda_p) W_n(\lambda - \lambda_p)$. See Priestley (1981, p. 581). A spectral window function is even, and so a spectral variance estimator $V_{\rm sf}(n)$ of the TAVC σ^2 can be taken as

$$V_{\rm sf}(n) = \frac{4\pi^2}{n} \sum_{p=-n/2}^{n/2} I_n(\lambda_p) W_n(\lambda_p). \tag{16}$$

The FFT algorithm allows for efficient computation of $d(\lambda_{-n/2})$, $\ldots$, $d(\lambda_0)$, $\ldots$, and $d(\lambda_{n/2})$, the finite Fourier transforms at these particular frequencies. One then computes $V_{\rm sf}(n)$ from (15) and (16).

By padding the vector of n values $X_r - \overline{Y}_{0,n}$ with $(n-1)$ zeros, the integral of (14) can be computed *exactly* from the $(2n-1)$ values $d(\lambda_p)$ ($\lambda_p = 2\pi p/(2n-1)$ for $p = 0, \pm 1, \ldots, \pm(n-1)$) instead of *approximated*. For simplicity, we will not do so, however.

The version of the FFT algorithm we consider is the one implemented in the C functions `four1` and `realft` of Press, Flannery, Teukolsky, and Vetterling (1988,pp. 411-412 and 417–418). One needs a first pass through the data in order to compute the sample mean $\overline{Y}_{0,n}$. The n values $(X_r - \overline{Y}_{0,n})$ are then stored in an array, called `data`, say. One also needs to store the n values of the finite Fourier transforms $(-n/2 < p \leq n/2)$. The algorithm will actually store them onto the same array `data`. To compute these values within a C program, one would call the function `realft`(data,n/2,-1), which in turn will call the

function `four1`(data,$n/2$,-1). The function `realft` requires of the order of n `flop`'s. The taxing computation is actually performed by `four1`. The so-called Danielson-Lanczos part of the function `four1` requires on the order of $n \log_2 n$ `flop`'s and the remainder of the function requires $O(n)$ `flop`'s (Cormen, Leiserson and Rivest 1990 p. 795). Therefore the dominant term for computing $V_{\rm sf}(n)$ is on the order of $n \log_2 n$ `flop`'s, and once again, we see that the effort required to compute the TAVC estimator is superlinear in the simulation run-length.

6- THE REGENERATIVE METHOD

For an overview of this method, see Shedler (1993). The process observations are divided up into regenerative cycles. Let the regeneration times be $1 = T_0, T_1, \ldots$, and let $\ell(n) = \sup\{k \geq 0 : T_k \leq n\}$ be the number of regenerative cycles completed by time n. For the ith regenerative cycle, let $Y_i = \sum_{j=T_{i-1}}^{T_i - 1} X_j$ be its total cost and $\tau_i = T_{i+1} - T_i$ be its length. Let $C_i = \sum_{j=1}^{i} Y_j$ be the total cost observed in the first i regenerative cycles.

After k cycles have been completed, the regenerative point estimator is given by $\alpha(T_k) = C_k/T_k$. The regenerative estimator for the variance is based on the fact that the TAVC can be written

$$\frac{E(Y_1 - \alpha\tau_1)^2}{E\tau_1} \tag{17}$$

and is given by

$$V_{\rm r}(n) = \frac{\sum_{j=1}^{\ell(n)} (Y_i - \alpha(T_{\ell(n)})\tau_i)^2}{T_{\ell(n)}}. \tag{18}$$

We will first evaluate the following two-pass algorithm for computing (18).

for $j = 1$ to $\ell(n)$, do
  calculate Y_j and τ_j
calculate $\alpha(T_{\ell(n)})$
calculate (18)

In determining the computational cost of this algorithm, we will assume zero cost for determining whether a regeneration occurred or not. This may be a good assumption in some contexts (e.g., simulating a discrete-time Markov chain on a countable state space, with regenerations defined as the hitting times of a distinguished state), or a poor assumption in others (e.g., for discrete-time Markov processes on a general state space, typically one needs to generate "splitting" random variables at each transition; see Glynn and L'Ecuyer (1995) for details). In any case, the check is linear in simulated time, and never superlinear.

The first step in the algorithm will take approximately $2n$ `flop`'s. The second will take approximately $2\ell(n)$ `flop`'s if one uses the Y_j's as intermediate quantities. The final step will require approximately $5\ell(n)$ `flop`'s, so that the expected total effort will be approximately $2n + 7\ell(n)$ `flop`'s. The algorithm will also require $2\ell(n)$ `stor`'s to record the Y_i's and τ_i's. But recall that $\ell(n)/n \to (E\tau_1)^{-1}$ as $n \to \infty$, so that the expected total computational effort will be approximately $(2 + 7(E\tau_1)^{-1})n$ `flop`'s.

A one pass algorithm that avoids the storage requirements of the above algorithm but is potentially numerically unstable was given in Shedler (1993). It is based on the observation that the TAVC (17) can also be written

$$\frac{\mathsf{Var}(Y_1) - 2\alpha\mathsf{Cov}(Y_1, \tau_1) + \alpha^2\mathsf{Var}(\tau_1)}{E\tau_1}.$$

Define the quantities $\bar{Y}_m = C_m/m, \bar{\tau}_m = T_m/m$,

$$\begin{aligned} S_{11}(m) &= \sum_{k=1}^{m} (Y_k - \bar{Y}_m)^2, \\ S_{22}(m) &= \sum_{k=1}^{m} (\tau_k - \bar{\tau}_m)^2, \text{ and} \\ S_{12}(m) &= \sum_{k=1}^{m} (Y_k - \bar{Y}_m)(\tau_k - \bar{\tau}_m). \end{aligned}$$

Note that $S_{11}(m)$ and $S_{22}(m)$ satisfy the recursion (5), and $S_{12}(m)$ satisfies the recursion

$$S_{12}(m) = S_{12}(m-1) + \frac{m-1}{m}(Y_m - \bar{Y}_{m-1})(\tau_m - \bar{\tau}_{m-1}). \tag{19}$$

The one pass algorithm for computing the estimator (18) is as follows.

set $S_{11}(1) = S_{12}(1) = S_{22}(1) = 0$
compute Y_1, τ_1. Set $C_1 = \bar{Y}_1 = Y_1$ and $T_1 = \bar{\tau}_1 = \tau_1$
for $m = 2$ to $\ell(n)$, do
  compute Y_m, τ_m
  compute $S_{11}(m)$, $S_{22}(m)$ from the recursion (5)
  compute $S_{12}(m)$ from the recursion (19)
  set $T_m = T_{m-1} + \tau_m$, $C_m = C_{m-1} + Y_m$
  set $\bar{\tau}_m = T_m/m$, $\bar{Y}_m = C_m/m$
compute $\alpha(n) = C_{\ell(n)}/n$
return $\dfrac{S_{11}(\ell(n)) - 2\alpha(n)S_{12}(\ell(n)) + \alpha(n)^2 S_{22}(\ell(n))}{(\ell(n) - 1)\bar{\tau}_{\ell(n)}}$.

It is easy to see that for a simulation run of length n (and hence $\ell(n)$ regenerative cycles), this algorithm will require approximately $(2 + 23/E\tau_1)n$ `flop`'s.

REFERENCES

Anderson, T. W. 1971. *Statistical analysis of time series.* New York: Wiley.

Bratley, P., B. L. Fox, and L. E. Schrage. 1987. *A guide to simulation,* 2nd Edition. New York: Springer-Verlag.

Carlstein, E. 1986. The use of subseries for estimating the variance of a general statistic from a stationary sequence. *Annals of Statistics* 14:1171–1179.

Chan, T. F., and J. G. Lewis. 1979. Computing standard deviations: Accuracy. *Communications of the ACM* 22:526–531.

Chan, T. F., G. H. Golub, and R. J. LeVeque. 1982. Updating formulae and a pairwise algorithm for computing sample variances. Compstat 1982, Proceedings of the 5th Symposium, eds. H.Caussinus, P. Ettinger, and J. R. Mathieu, 30–41. Cambridge, Massachusetts: Physica-Verlag.

Chan, T. F., G. H. Golub, and R. J. LeVeque. 1983. Algorithms for computing the sample variance: analysis and recommendations. *The American Statistician* 37:242–247.

Clark, G. M. 1980. Recursive estimation of the variance of the sample average. *ACM Transactions on Mathematical Software* 6:58–67.

Cormen, T. H., C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to algorithms.* Cambridge, Massachusetts: The MIT Press.

Damerdji, H. 1991. Strong consistency and other properties of the spectral variance estimator. *Management Science* 37:1424–1440.

Damerdji, H. 1994. Strong consistency of the variance estimator in steady-state simulation output analysis. *Mathematics of Operations Research* 19:494–512.

Glynn, P. W., and P. L'Ecuyer. 1995. Likelihood ratio gradient estimation for stochastic recursions. *Advances in Applied Probability* 27:1019–1053.

Glynn, P. W., and W. Whitt. 1991. Estimating the asymptotic variance with batch means. *Operations Research Letters* 10:431–435.

Glynn, P. W., and W. Whitt. 1992. The asymptotic efficiency of simulation estimators. *Operations Research* 40:505–520.

Goldsman, D., and M. S. Meketon. 1986. A comparison of several variance estimators. Technical report J-85-12, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Hanson, R. J. 1975. Stably updating mean and standard deviation of data. *Communications of the ACM* 18:57–58.

Higham, N. J. 1993. The accuracy of floating point summation. *SIAM Journal of Scientific Computation* 14:783–799.

Law, A. M., and W. D. Kelton. 1991. *Simulation modeling & analysis,* 2nd edition. New York: McGraw Hill.

Meketon, M., and B. W. Schmeiser. 1984. Overlapping batch means: Something for nothing? In *Proceedings of the 1984 Winter Simulation Conference*, eds. S. Sheppard, U. W. Pooch, and C. D. Pegden, 227–230. IEEE, Piscataway, New Jersey.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1988. *Numerical recipes in C, The art of scientific computing.* Cambridge, England: Cambridge University Press.

Priestley, M. B. 1981. *Spectral analysis and time series.* New York: Academic Press.

Shedler, G. S. 1993. *Regenerative stochastic simulation.* San Diego: Academic Press.

Song W. T., and B. W. Schmeiser. 1995. Optimal mean-squared-error batch sizes. *Management Science* 41:110–123.

West, D. H. D. 1979. Updating mean and variance estimates: *Communications of the ACM* 22:532–535.

Youngs, E. A., and E. M. Cramer. 1971. Some results relevant to choice of sum and sum-of-product algorithms. *Technometrics* 13:657–665.

AUTHOR BIOGRAPHIES

HALIM DAMERDJI is an assistant professor in the Department of Industrial Engineering at North Carolina State University. He received a Ph.D. degree in industrial engineering from the University of Wisconsin-Madison. He has held positions at the Ecole Nationale Polytechnique of Algiers and Purdue University. His research interests are in simulation and applied stochastic processes.

SHANE G. HENDERSON graduated from the Department of Operations Research at Stanford University in 1996. He then joined the Department of Industrial and Operations Engineering at the University of Michigan at Ann Arbor. His research interests include discrete-event simulation, Markov processes and queueing theory.

PETER W. GLYNN received his Ph.D. from Stanford University, after which he joined the faculty of the Department of Industrial Engineering at the University of Wisconsin-Madison. In 1987, he returned to Stanford, where he currently holds the Thomas Ford Chair in the Department of Engineering-Economic Systems and Operations Research. He was a co-winner of the 1993 Outstanding Simulation Publication Award sponsored by the TIMS College on Simulation. His research interests include discrete-event simulation, computational probability, queueing, and general theory for stochastic systems.