Two-Timescale Simultaneous Perturbation Stochastic Approximation Using Deterministic Perturbation Sequences

SHALABH BHATNAGAR
Indian Institute of Science, Bangalore
MICHAEL C. FU and STEVEN I. MARCUS
University of Maryland, College Park, Maryland
and
I-JENG WANG
Johns Hopkins University, Applied Physics Laboratory, Laurel, Maryland

Simultaneous perturbation stochastic approximation (SPSA) algorithms have been found to be very effective for high-dimensional simulation optimization problems. The main idea is to estimate the gradient using simulation output performance measures at only *two* settings of the N-dimensional parameter vector being optimized rather than at the $N + 1$ or $2N$ settings required by the usual one-sided or symmetric difference estimates, respectively. The two settings of the parameter vector are obtained by *simultaneously* changing the parameter vector in each component direction using *random* perturbations. In this article, in order to enhance the convergence of these algorithms, we consider deterministic sequences of perturbations for two-timescale SPSA algorithms. Two constructions for the perturbation sequences are considered: complete lexicographical cycles and much shorter sequences based on normalized Hadamard matrices. Recently, one-simulation versions of SPSA have been proposed, and we also investigate these algorithms using deterministic sequences. Rigorous convergence analyses for all proposed algorithms are presented in detail.

S. Bhatnagar was partially supported in his research through a fellowship at the Free University, Amsterdam during 2000–2001, and thanks Professor G. Koole for his hospitality during his stay.
The work of M. Fu and S. Marcus was supported in part by the National Science Foundation (NSF) under Grant DMI-9988867, and by the the Air Force Office of Scientific Research under Grant F496200110161.
The work of I.-J. Wang was supported in part by JHU/APL Independent Research and Development Program.
Authors' addresses: S. Bhatnagar, Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India; email: shalabh@csa.iisc.ernet.in; M. C. Fu, The Robert H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park, MD 20742; email: mfu@isr.umd.edu; S. I. Marcus, Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742; email: marcus@eng.umd.edu; I-J.Wang, Johns Hopkins University, Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, MD 20723; email I-Jeng.Wang@jhuapl.edu.

Extensive numerical experiments on a network of $M/G/1$ queues with feedback indicate that the deterministic sequence SPSA algorithms perform significantly better than the corresponding randomized algorithms.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: *Probabilistic Algorithms (including Monte Carlo)*; I.6.0 [**Simulation and Modeling**]: General; I.6.1 [**Simulation and Modeling**]: Simulation Theory

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Simulation optimization, stochastic approximation, SPSA, two-timescale algorithms, deterministic perturbations, Hadamard matrices

1. INTRODUCTION

Simultaneous perturbation stochastic approximation (SPSA), introduced in Spall [1992], has attracted considerable attention in recent years because of its generality and efficiency in addressing high-dimensional stochastic optimization problems (see, e.g., Bhatnagar et al. [2001a], Chen et al. [1999], Fu and Hill [1997], Gerencsér [1999], and Kleinman et al. [1999]). Like algorithms based on finite differences Kiefer–Wolfowitz stochastic approximation, SPSA requires only estimates of the objective function itself. On the other hand, algorithms based on Robbins-Monro stochastic approximation, which employ direct gradient estimators using for example perturbation analysis generally require additional knowledge on the underlying structure of the system being optimized. However, the one-sided and two-sided finite difference Kiefer–Wolfowitz algorithms (Kushner and Clark [1978]) require $N + 1$ and $2N$ samples of the objective function, respectively, whereas SPSA typically uses only two samples for (any) N-dimensional parameter, updating the entire parameter vector at each update epoch along an estimated gradient direction obtained by *simultaneously* perturbing all parameter components in *random* directions. SPSA, like traditional stochastic approximation, was originally designed for continuous-parameter optimization problems, although modifications for discrete optimization have been proposed recently by Gerencsér et al. [1999]. In this article, we address only the continuous parameter setting.

Using SPSA, one obtains in effect the correct (steepest descent) direction because of the form of the gradient estimate, where the random perturbations are chosen to be mean-zero and mutually independent (most commonly generated by using independent, symmetric, Bernoulli distributed random variables). As a result of this choice, the estimate along 'undesirable gradient directions' averages to zero. The main driving force behind the work reported here is that this averaging can also be achieved in a less "noisy" fashion using *deterministic* perturbation sequences, similar in spirit to the use of quasi-Monte Carlo sequences as in Niederreiter [1992, 1995], in place of pseudorandom numbers for numerical integration. We have found that in certain scenarios, deterministic perturbations are theoretically sound and lead to faster convergence in our numerical experiments. Deterministic perturbations in random direction Kiefer-Wolfowitz (RDKW) algorithms have also been studied in Sandilya and

Kulkarni [1997]. In Wang and Chong [1998], it is shown that RDKW and SPSA algorithms have similar asymptotic performance.

We propose two different constructions of deterministic perturbation sequences. The first sequence follows a lexicographic ordering to cyclically visit exactly half (2^{N-1}) of all the 2^N points in the space of perturbations (note that the other half are automatically visited by being the mirror image). The second sequence is constructed based on normalized Hadamard matrices [Hadamard 1893; Seberry and Yamada 1992], which drastically reduces the required number of points to be visited cyclically in the perturbation space to $2^{\lceil \log_2 N \rceil}$ points. The idea behind the Hadamard matrix construction is to reduce the contribution of aggregate bias over iterations. We prove convergence for both constructions by directly imposing the desired properties on the structure of the sequence of perturbations.

In the following, we introduce two-timescale SPSA algorithms and examine several forms of these in a simulation setting, by varying the number of simulations, the nature of perturbations, and the algorithm type depending on the nature of update epochs of the algorithm. The rest of the article is organized as follows. In the next section, we formulate the problem, describe the various algorithms, present the first (lexicographic) construction for deterministic perturbations and present the convergence analysis of these algorithms. In Section 3, we present the algorithms with deterministic perturbations using normalized Hadamard matrices, present the relevant results and show that the convergence analysis of the previous algorithms carries over for these algorithms. We present the results of numerical experiments in Section 4. Finally, Section 5 provides concluding remarks and some avenues for further research.

2. ALGORITHMS WITH LEXICOGRAPHICALLY ORDERED DETERMINISTIC PERTURBATIONS

The standard stochastic approximation algorithm is of the following type:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + a_k Y_k, \tag{1}$$

where $\hat{\theta}_k \equiv (\hat{\theta}_{k,1}, \ldots, \hat{\theta}_{k,N})^T$ are tunable parameter vectors, a_k correspond to step-sizes and $Y_k \equiv (Y_{k,1}, \ldots, Y_{k,N})^T$ are certain 'criterion functions'. The Robbins–Monro algorithm, for instance, finds zeroes of a function $F(\theta)$, given sample observations $f(\theta, \xi)$ with random noise ξ such that $F(\theta) = E[f(\theta, \xi)]$. The Robbins–Monro algorithm therefore corresponds to $Y_{k,i} = f(\hat{\theta}_{k,i}, \xi_{k,i})$ in (1), where $\xi_{k,i}, i = 1, \ldots, N, k \geq 0$, are independent and identically distributed (i.i.d.) random vectors. On the other hand, the Kiefer–Wolfowitz algorithm tries to minimize the objective function $F(\theta)$ by finding zeroes of $\nabla F(\theta)$ using a gradient descent algorithm, that is, in (1)

$$Y_{k,i} = -\left(\frac{f(\hat{\theta}_k + \delta_k e_i, \xi^+_{k,i}) - f(\hat{\theta}_k - \delta_k e_i, \xi^-_{k,i})}{2\delta_k} \right), \tag{2}$$

where $\delta_k \to 0$ as $k \to \infty$ 'slowly enough', $\xi^+_{k,i}, \xi^-_{k,i}$ are independent and identically distributed, and e_i is the unit vector with 1 in the ith place and 0's elsewhere. From (1), it is easy to see that for getting the full estimate $Y_k \equiv (Y_{k,1}, \ldots, Y_{k,N})^T$

at any update epoch k, one needs $2N$ samples of the objective function $f(\cdot,\cdot)$. This can be reduced to $(N+1)$ samples if a one-sided estimate is used instead. In contrast, SPSA requires only two samples of the objective function at each update epoch, as follows:

$$Y_{k,i} = -\left(\frac{f(\hat{\theta}_k + \delta_k \triangle(k), \xi_k^+) - f(\hat{\theta}_k - \delta_k \triangle(k), \xi_k^-)}{2\delta_k \triangle_i(k)}\right), \tag{3}$$

where ξ_k^+, ξ_k^- are independent and identically distributed random vectors in an appropriate Euclidean space, $\triangle(k) \equiv (\triangle_1(k), \ldots, \triangle_N(k))^T$, where $\triangle_i(k), i = 1, \ldots, N, k \geq 0$, are independent and identically distributed, zero-mean random variables with $P(\triangle_i(k) = 0) = 0$. In many applications, these are chosen to be independent and identically distributed Bernoulli distributed random variables with $P(\triangle_i(k) = +1) = P(\triangle_i(k) = -1) = 1/2$. SPSA has been a very popular algorithm for function minimization mainly since it requires only two estimates of the objective function at each step of the iteration.

More recently, another form of SPSA that requires only one sample of the objective function has also been studied in Spall [1997] and Spall and Cristion [1998]. In this,

$$Y_{k,i} = -\left(\frac{f(\hat{\theta}_k + \delta_k \triangle(k))}{\delta_k \triangle_i(k)}\right), \tag{4}$$

or alternatively,

$$Y_{k,i} = \left(\frac{f(\hat{\theta}_k - \delta_k \triangle(k))}{\delta_k \triangle_i(k)}\right). \tag{5}$$

However, it was observed that the one-sample form has significant 'extra bias' in comparison to the two-sample form (3), so that in general the latter is preferable. SPSA is supported by considerable theory on its convergence properties, and this theory is based (in part) on using random perturbations to compute the gradient approximation. In a large class of important applications, the objective function is estimated by running a simulation, so that the two-sample SPSA algorithm, for instance, would require two parallel simulations.

We now introduce our framework. Let $\{X_n, n \geq 1\}$ be an $\mathcal{R}^d$-valued (for some given $d \geq 1$) parameterized Markov process with a tunable N-dimensional parameter θ that takes values in a compact set $C \subset \mathcal{R}^N$. We assume in particular C to be of the form $C \stackrel{\triangle}{=} \prod_{i=1}^{N} [\theta_{i,\min}, \theta_{i,\max}]$. Note that we constrain our algorithms to evolve within the set C by using projection operators. In general, the set C could have other forms as well, for example, a disjoint union of compact intervals. The projection operator could then be chosen such that it projects the iterates onto the set from which it is the closest, so as to ignore (say) undesirable points that may lie in between these subsets. Let $h : \mathcal{R}^d \to \mathcal{R}^+$ be a given bounded and continuous cost function. Our aim is to find a θ that minimizes the long-run average cost

$$J(\theta) = \lim_{l \to \infty} \frac{1}{l} \sum_{j=0}^{l-1} h(X_j). \tag{6}$$

Thus, one needs to evaluate $\nabla J(\theta) \equiv (\nabla_1 J(\theta), \ldots, \nabla_N J(\theta))^T$. Consider $\{X_j\}$ governed by parameter θ, and N other sequences $\{X_j^i\}$, $i = 1, \ldots, N$, each governed by $\theta + \delta e_i$, $i = 1, \ldots, N$, respectively. Then

$$\nabla_i J(\theta) = \lim_{\delta \to 0} \left(\frac{1}{\delta} \lim_{l \to \infty} \frac{1}{l} \sum_{j=0}^{l-1} (h(X_j^i) - h(X_j)) \right). \tag{7}$$

The sequences $\{X_j\}$, $\{X_j^i\}$, $i = 1, \ldots, N$, correspond to $(N + 1)$ parallel simulations. One can also consider the gradient (7) in a form (similar to (2)) with $2N$ parallel simulations. However, in any of these forms, it is clear (cf. (7)) that the outer limit is taken only after the inner limit. For classes of systems for which the two limits in (7) can be interchanged, gradient estimates from a single sample path can be obtained using infinitesimal perturbation analysis (IPA) [Chong and Ramadge 1993, 1994; Fu 1990; Ho and Cao 1991], which when applicable is usually more efficient than finite differences. Since SPSA does not require the interchange, it is more widely applicable. Broadly, any recursive algorithm that computes optimum θ based on (7) should have two loops, the outer loop (corresponding to parameter updates) being updated once after the inner loop (corresponding to data averages) has converged. Thus, one can physically identify separate timescales, the faster one on which data based on a fixed parameter value is aggregated and averaged, and the slower one on which parameter is updated once the averaging is complete for one latter iteration. The same effect as from different physical timescales can also be achieved by using different step-size schedules (also called timescales) in the stochastic approximation algorithm. Based on this, algorithms were proposed in Bhatnagar and Borkar [1997, 1998] that require $(N+1)$ parallel simulations. In Bhatnagar et al. [2001a], the SPSA versions of these algorithms SPSA1-2R and SPSA2-2R, respectively, were presented in the setting of hidden Markov models. Both the SPSA versions require only two parallel simulations, leading to significant performance improvements over the corresponding algorithms of Bhatnagar and Borkar [1997, 1998]. In Bhatnagar et al. [2001b], algorithm SPSA1-2R was applied to a problem of closed loop feedback control in available bit rate (ABR) service in asynchronous transfer mode (ATM) networks.

In this article, we investigate the use of deterministic perturbation sequences, that is, those sequences $\{\triangle(n)\}$ of perturbations that cyclically pass through a set of points in the perturbation space in a deterministic fashion. We present two constructions based on deterministic perturbations—lexicographic and normalized Hadamard matrix based. In what follows, we shall develop ten SPSA algorithms, all using two timescales based on two step-size sequences $\{a(n)\}$ and $\{b(n)\}$ (as opposed to the one timescale version implied by the traditional form of SA given by (1)). We use the following general notation for these algorithms. SPSAi-jK is the two-timescale SPSA algorithm of type i ($i = 1, 2$), that uses j simulations ($j = 1, 2$). Also, K=R signifies that the particular algorithm uses randomized difference perturbations as in regular SPSA, whereas K=L (respectively, H) signifies that the particular algorithm uses deterministic perturbations based on lexicographical (respectively normalized Hadamard matrix based) ordering of points in the perturbation space. Algorithms of type

1 correspond to those having similar *structure* as SPSA1 of Bhatnagar et al. [2001a] (SPSA1-2R here), whereas those of type 2 correspond to SPSA2 of Bhatnagar et al. [2001a] (SPSA2-2R here). Algorithms of type 1 (SPSA1-2R, SPSA1-2L, SPSA1-2H, SPSA1-1R, SPSA1-1L and SPSA1-1H) update the parameter at instants n_m defined by $n_0 = 1$ and

$$n_m = \min\{j > n_{m-1} \mid \sum_{i=n_{m-1}+1}^{j} a(i) \geq b(m)\}, \ m \geq 1.$$

It was shown in Bhatnagar and Borkar [1997] that this sequence $\{n_m\}$ increases exponentially. On the other hand, algorithms of type 2 (SPSA2-2R, SPSA2-2L, SPSA2-2H, SPSA2-1R, SPSA2-1L and SPSA2-1H) update the parameter once every fixed L instants. In the corresponding algorithm of Bhatnagar and Borkar [1998] that requires $(N + 1)$ parallel simulations, $L = 1$, that is, the updates are performed at each instant. However in the SPSA version of the same in Bhatnagar et al. [2001a], it was observed that an extra averaging in addition to the two timescale averaging is required to ensure good algorithmic behavior when the parameter dimension is high. Thus, algorithms of type 1 tend to be more in the spirit of physical timescale separation wherein parameter updates (after a while) are performed after data gets averaged over long (and increasing) periods of time, whereas those of type 2 update the parameter once every fixed number of time instants by explicitly using different timescales of the stochastic approximation algorithm through coupled recursions that proceed at "different speeds." In systems that have a significantly high computational requirement, algorithms of type 2 may be computationally superior because of faster update epochs. Computational comparisons of the randomized algorithms SPSA1-2R and SPSA2-2R have been provided in Bhatnagar et al. [2001a].

We make the following assumptions:

(A1) The basic underlying process $\{X_n, n \geq 1\}$ is ergodic Markov for each fixed θ.

(A2) The long run average cost $J(\theta)$ is continuously differentiable with bounded second derivative.

(A3) The step-size sequences $\{a(n)\}$ and $\{b(n)\}$ are defined by

$$a(0) = \hat{a}, b(0) = \hat{b}, a(n) = \hat{a}/n, b(n) = \hat{b}/n^{\alpha},$$
$$n \geq 1, \frac{1}{2} < \alpha < 1, 0 < \hat{a}, \hat{b} < \infty. \tag{8}$$

We now provide the motivation behind these assumptions. First, (A1) ensures that the limit in the expression for $J(\theta)$ given by (6) is well defined. Also, $\{X_n\}$ governed by a time varying parameter sequence $\{\theta_n\}$ that is updated according to any of the algorithms that we present in this paper, would continue to be Markov. Further, (A1) implies that for any fixed θ, starting from any initial distribution, the distribution of the process $\{X_n\}$ shall converge to the stationary distribution under θ. One can also argue that the overall process $\{X_n\}$ governed by $\{\theta_n\}$ with θ_n updated according to any of our algorithms, continues to be stable. (A1) is thus also used in deriving the asymptotic equivalence of the

two-simulation (respectively, one-simulation) algorithms with Eq. (12) (respectively, Eq. (27)) (cf. Bhatnagar et al. [2001a]).

Assumption (A2) is a technical requirement that is needed to push through a Taylor series argument to essentially show that SPSA gives the correct gradient descent directions on the average. Moreover, $J(\theta)$ also serves as an associated Liapunov function for the ODE (18) and is therefore required to be continuously differentiable.

The step-size sequences in (A3) are of the standard type in stochastic approximation algorithms, satisfying the essential properties

$$\sum_{n=1}^{\infty} a(n) = \sum_{n=1}^{\infty} b(n) = \infty, \quad \sum_{n=1}^{\infty} a(n)^2, \sum_{n=1}^{\infty} b(n)^2 < \infty, \tag{9}$$

$$a(n) = o(b(n)). \tag{10}$$

Intuitively, (10) means that $\{b(n)\}$ corresponds to the faster timescale and $\{a(n)\}$ to the slower one, since $\{a(n)\}$ goes to zero faster than $\{b(n)\}$ does.

Let $\delta > 0$ be a fixed small constant, let $\pi_i(x) \stackrel{\triangle}{=} \min(\max(\theta_{i,\min}, x), \theta_{i,\max})$, $i = 1, \ldots, N$, denote the point closest to $x \in \mathcal{R}$ in the interval $[\theta_{i,\min}, \theta_{i,\max}] \subset \mathcal{R}, i = 1, \ldots, N$, and let $\pi(\theta)$ denote $\pi(\theta) \stackrel{\triangle}{=} (\pi_1(\theta_1), \ldots, \pi_N(\theta_N))^T$ for $\theta = (\theta_1, \ldots, \theta_N)^T \in \mathcal{R}^N$. Then $\pi(\theta)$ is a projection of θ on to the set C. We now recall the algorithm SPSA1-2R of Bhatnagar et al. [2001a].

2.1 Algorithm SPSA1-2R

We assume for simplicity that $\triangle_k(m)$, for all $k = 1, \ldots, N$, and integers $m \geq 0$, are mutually independent, Bernoulli distributed random variables taking values ± 1, with $P(\triangle_k(m) = +1) = P(\triangle_k(m) = -1) = 1/2$. Let $\triangle(m) \stackrel{\triangle}{=} (\triangle_1(m), \ldots, \triangle_N(m))^T$ represent the vector of the random variables $\triangle_1(m), \ldots, \triangle_N(m)$. The perturbation sequences can also have more general distributions that satisfy Condition (B) (see Bhatnagar et al. [2001a]) below.

Condition (B). There exists a constant $\bar{K} < \infty$, such that for any $l \geq 0$, and $i \in \{1, \ldots, N\}$, $E[(\triangle_i(l))^{-2}] \leq \bar{K}$.

Minor variants of this condition are for instance available in Spall [1992]. In this article, however, we consider only perturbation sequences formed from independent and identically distributed, Bernoulli distributed random variables for the sake of simplicity.

SPSA1-2R

Consider two parallel simulations $\{X_j^k\}$, $k = 1, 2$, respectively, governed by $\{\tilde{\theta}_j^k\}$, $k = 1, 2$, as follows: For the process $\{X_j^1\}$, we define $\tilde{\theta}_j^1 = \theta(m) - \delta\triangle(m)$, for $n_m < j \leq n_{m+1}$, $m \geq 0$. The parameter sequence $\{\tilde{\theta}_j^2\}$ for $\{X_j^2\}$ is similarly defined by $\tilde{\theta}_j^2 = \theta(m) + \delta\triangle(m)$, for $n_m < j \leq n_{m+1}$, $m \geq 0$. Here,

$\theta(m) \stackrel{\triangle}{=} (\theta_1(m), \ldots, \theta_N(m))^T$ is the value of the parameter update that is governed by the following recursion equations. For $i = 1, \ldots, N$,

$$\theta_i(m+1) = \pi_i \left(\theta_i(m) + \sum_{j=n_m+1}^{n_{m+1}} a(j) \left(\frac{h(X_j^1) - h(X_j^2)}{2\delta \triangle_i(m)} \right) \right), \tag{11}$$

$m \geq 0$. Thus, we hold $\theta(m)$ and $\triangle(m)$ fixed over intervals $n_m < j \leq n_{m+1}$, $m \geq 0$, for the two simulations and at the end of these intervals, update $\theta(m)$ according to (11), and generate new samples for each of the $\triangle_i(m)$.

Next, we consider perturbation sequences $\{\triangle(m)\}$ that are generated deterministically for the algorithm SPSA1-2L. We consider first a lexicographic construction for generating these sequences.

2.2 Lexicographical Deterministic Perturbations and Algorithm SPSA1-2L

When the perturbations are randomly generated, unbiasedness is achieved by way of expectation and mutual independence. The use of deterministic perturbation sequences allows a construction that can guarantee the desirable averaging properties over a cycle of the sequence. In this and other constructions, we will assume that as in the random perturbation case, where a symmetric Bernoulli distribution was used, each component of the N-dimensional perturbation vector takes on a value ± 1; thus, the resulting discrete set $E = \{\pm 1\}^N$ on which the vector takes values has cardinality 2^N. In this section, we will consider a complete cyclical sequence on the set of possible perturbations using the natural lexicographical ordering. In Section 3, we consider a much more compact sequence using normalized Hadamard matrices.

Fix $\triangle_1(m) = -1 \; \forall m \geq 0$, and lexicographically order all points in the discrete set $F \subset E$ with $F = \{e_0, \ldots, e_{2^{N-1}-1}\}$. Thus, F corresponds to the resulting set in which $\triangle(m)$ takes values. Next, we set $\triangle(0) = e_0$ and move the sequence $\{\triangle(m)\}$ cyclically through the set F by setting $\triangle(m) = e_{s_m}$, where $s_m \stackrel{\triangle}{=} (m - [m/2^{N-1}]2^{N-1})$ is the remainder from the division $m/2^{N-1}$. Thus, for $\theta = (\theta_1, \theta_2, \theta_3)^T$, the elements of the set F are ordered as follows: $f_0 = (-1, -1, -1)^T$, $f_1 = (-1, -1, +1)^T$, $f_2 = (-1, +1, -1)^T$ and $f_3 = (-1, +1, +1)^T$. We now set $\triangle(0) = f_0$ and move the sequence cyclically through all points $f_0, \ldots, f_3$. Note that in this manner, for two-simulation algorithms, we only require exactly half the number of points in the space of perturbations. It will be shown later that the bias in two-simulation deterministic perturbation algorithms contains perturbation ratios of the type $\triangle_j(m)/\triangle_i(m)$, $j \neq i$, $i, j \in \{1, \ldots, N\}$ as components. The corresponding one-simulation algorithms contains both terms of the type $\triangle_j(m)/\triangle_i(m)$, $j \neq i, i, j \in \{1, \ldots, N\}$, as well as those of type $1/\triangle_i(m), i \in \{1, \ldots, N\}$. A similar construction for perturbation sequences in one-simulation algorithms therefore requires that the perturbation sequence pass through the full set of points E and not just the half obtained by holding $\triangle_1(m)$ to a fixed value as above.

Table I. Perturbation Ratios and Inverses for $N = 3$ using Lexicographic Construction

$j = 8m+l$, $m \geq 0$	Point $(\triangle(j) = e_l)$	$\frac{\triangle_1(j)}{\triangle_2(j)}$	$\frac{\triangle_1(j)}{\triangle_3(j)}$	$\frac{\triangle_2(j)}{\triangle_3(j)}$	$\frac{1}{\triangle_1(j)}$	$\frac{1}{\triangle_2(j)}$	$\frac{1}{\triangle_3(j)}$
$8m$	$(-1,-1,-1)^T$	$+1$	$+1$	$+1$	-1	-1	-1
$8m+1$	$(-1,-1,+1)^T$	$+1$	-1	-1	-1	-1	$+1$
$8m+2$	$(-1,+1,-1)^T$	-1	$+1$	-1	-1	$+1$	-1
$8m+3$	$(-1,+1,+1)^T$	-1	-1	$+1$	-1	$+1$	$+1$
$8m+4$	$(+1,-1,-1)^T$	-1	-1	$+1$	$+1$	-1	-1
$8m+5$	$(+1,-1,+1)^T$	-1	$+1$	-1	$+1$	-1	$+1$
$8m+6$	$(+1,+1,-1)^T$	$+1$	-1	-1	$+1$	$+1$	-1
$8m+7$	$(+1,+1,+1)^T$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$

For $N = 3$, we describe in Table I the ratios of quantities of the type $1/\triangle_i(j)$, $i = 1,2,3$, and $\triangle_k(j)/\triangle_i(j)$, $k \neq i$, respectively. From Table I, it is easy to see that $\sum_{j=0}^{n} \triangle_1(j)/\triangle_2(j)$ equals zero for any $n = 4m-1, m \geq 1$. Also $\sum_{j=0}^{n} \triangle_1(j)/\triangle_3(j)$ and $\sum_{j=0}^{n} \triangle_2(j)/\triangle_3(j)$ equal zero for any $n = 2m-1, m \geq 1$. Thus (for $N = 3$), all quantities of the type $\sum_{j=0}^{n} \triangle_k(j)/\triangle_i(j), i \neq k$, become zero at least once in every four iterations. Also note in this example that by forming vectors $R(j) \stackrel{\triangle}{=} (\triangle_1(j)/\triangle_2(j), \triangle_1(j)/\triangle_3(j), \triangle_2(j)/\triangle_3(j))^T$, one finds for any $m \geq 0, j \in \{0,1,\ldots,7\}, R(8m+j) = R(8m+7-j)$. We point out that the analysis for two-simulation algorithms would work equally well with deterministic perturbation sequences that are similarly defined on the set $E \backslash F$ (i.e., the set obtained by holding $\triangle_1(m) = +1$ for all m, instead of -1).

SPSA1-2L

The algorithm SPSA1-2L is exactly the same as algorithm SPSA1-2R but with lexicographical deterministic perturbation sequences $\{\triangle(m)\}$ generated according to the procedure just described.

Next, we provide the convergence analysis for SPSA1-2L.

2.3 Convergence Analysis of SPSA1-2L

It was shown using ordinary differential equation (ODE) analysis in Bhatnagar et al. [2001a], that the randomized difference algorithm SPSA1-2R asymptotically tracks the stable points of the ODE (18). As an initial step, it was shown that under (A1) SPSA1-2R is asymptotically equivalent to the following algorithm, in the sense that the differences between the gradient estimates in these algorithms is $o(1)$:

$$\theta_i(m+1) = \pi_i\left(\theta_i(m) + b(m)\left(\frac{J(\theta(m) - \delta\triangle(m)) - J(\theta(m) + \delta\triangle(m))}{2\delta\triangle_i(m)}\right)\right), \quad (12)$$

for $i = 1,\ldots,N, m \geq 0$. SPSA1-2R can therefore be treated as being the same as (12), except for additional asymptotically diminishing error terms on the RHS of (12), which can again be taken care of in a routine manner (see Borkar [1997]). One can similarly show as in Bhatnagar et al. [2001a] that SPSA1-2L with deterministic perturbations $\{\triangle(m)\}$ is also asymptotically equivalent under (A1) to (12). We have the following basic result.

THEOREM 2.1. *The deterministic perturbations* $\triangle(n) \stackrel{\triangle}{=} (\triangle_1(n), \ldots, \triangle_N(n))^T$ *satisfy:*

$$\sum_{n=s}^{s+2^{N-1}-1} \frac{\triangle_i(n)}{\triangle_j(n)} = 0 \text{ for any } s \geq 0,\ i \neq j,\ i, j \in \{1, \ldots, N\}.$$

PROOF. We first show that

$$\sum_{n=2^{N-1}m}^{2^{N-1}m+2^{N-1}-1} \frac{\triangle_i(n)}{\triangle_j(n)} = 0$$

for any $m \geq 0$, $i \neq j$, $i, j, k \in \{1, \ldots, N\}$. Note that $\triangle_1(n) = -1\ \forall n$. For $k \in \{2, \ldots, N\}$, one can write

$$\begin{aligned} \triangle_k(n) \;=\; & I\{n = 2^{N-k}m_k + l_k, m_k \geq 0, l_k \in \{2^{N-k-1}, \ldots, 2^{N-k} - 1\}\} \\ & - I\{n = 2^{N-k}m_k + l_k, m_k \geq 0, l_k \in \{0, 1, \ldots, 2^{N-k-1} - 1\}\}, \end{aligned} \tag{13}$$

where $I\{\cdot\}$ represents the characteristic or indicator function. Here, we use the fact that given n and k, there exist unique integers $m_k \geq 0$ and $l_k \in \{0, \ldots, 2^{N-k} - 1\}$ such that $n = 2^{N-k}m_k + l_k$. Thus, in (13), we merely split the full set in which l_k takes values into two disjoint subsets, one on which $\triangle_k(n) = -1$ and the other on which $\triangle_k(n) = +1$. In other words, we have described $\triangle_k(n)$ solely on the basis of 'cycles' after which these change sign. The largest cycle amongst components $k = 2, \ldots, N$, corresponds to $\triangle_2(n)$ for which n is represented as $n = 2^{N-1}m_2 + l_2$. In what follows, we will first uniformly write all components in terms of the largest cycle.

Note that because of the lexicographical ordering, the range of each $\triangle_k(n)$, $k = 2, \ldots, N$, can be split into blocks of 2^{N-k} of -1 and $+1$ elements over which these sum to zero, that is,

$$\sum_{n=2^{N-k}m}^{2^{N-k}(m+1)-1} \triangle_k(n) = 0.$$

The total number of such blocks given N (the dimension of the vector) and k is $2^{N-1}/2^{N-k} = 2^{k-1}$. Thus, for any $k = 2, \ldots, N$, $\triangle_k(n)$ can be written as

$$\begin{aligned} \triangle_k(n) \;=\; & I\{n = 2^{N-1}m + l, m \geq 0, l \in \{2^{N-k}, \ldots, 2 \times 2^{N-k} - 1\} \\ & \cup\ \{3 \times 2^{N-k}, \ldots, 4 \times 2^{N-k} - 1\} \\ & \cup \ldots \cup \{(2^{k-1} - 1)2^{N-k}, \ldots, 2^{k-1}2^{N-k} - 1\}\} \\ & -I\{n = 2^{N}m + l, m \geq 0, l \in \{0, \ldots, 2^{N-k} - 1\} \\ & \cup\ \{2 \times 2^{N-k}, \ldots, 3 \times 2^{N-k} - 1\} \\ & \cup \ldots \cup \{(2^{k-1} - 2)2^{N-k}, \ldots, (2^{k-1} - 1)2^{N-k} - 1\}\}, \end{aligned} \tag{14}$$

where we have

$$\frac{\triangle_k(n)}{\triangle_l(n)} = \frac{\triangle_l(n)}{\triangle_k(n)}, \quad k \neq l.$$

Thus, without loss of generality assume $k < l$.

For $\triangle_k(n)$, the space $\{0, \ldots, 2^{N-1}-1\}$ is partitioned into 2^{k-1} subsets of type

$$A_{r,k} \stackrel{\triangle}{=} \{r \times 2^{N-k}, \ldots, (r+1)2^{N-k}-1\}, \quad r \in \{0, 1, \ldots, 2^{k-1}-1\},$$

with each subset containing 2^{N-k} elements. For $\triangle_l(n)$, the space $\{0, \ldots, 2^{N-1}-1\}$ is similarly partitioned (as above) into 2^{l-1} subsets of the type

$$A_{\bar{r},l} \stackrel{\triangle}{=} \{\bar{r} \times 2^{N-l}, \ldots, (\bar{r}+1)2^{N-l}-1\}, \quad \bar{r} \in \{0, 1, \ldots, 2^{l-1}-1\}.$$

Thus, since $k < l$, $2^k < 2^l$ and so $2^{N-k} > 2^{N-l}$. Also the number of elements in the partition $A_k \stackrel{\triangle}{=} \{A_{r,k}\}$ equals 2^k, while those in partition $A_l \stackrel{\triangle}{=} \{A_{\bar{r},l}\}$ equal 2^l. From the construction, it is clear that the partition sets $A_{r,k}$ can be derived from those of $A_{\bar{r},l}$ by taking union over 2^{l-k} successive sets in $A_{\bar{r},l}$. Thus,

$$\begin{aligned} A_{r,k} &= \{r \times 2^{N-k}, \ldots, (r+1)2^{N-k}-1\} \\ &= \{r \times 2^{N-k}, \ldots, r \times 2^{N-k} + 2^{N-l} - 1\} \\ &\cup \{r \times 2^{N-k} + 2^{N-l}, \ldots, r \times 2^{N-k} + 2^{N-l+1} - 1\} \\ &\cup \ldots \cup \{(r+1)2^{N-k} - 2^{N-l}, \ldots, (r+1)2^{N-k} - 1\}. \end{aligned}$$

By construction now, $\triangle_l(n) = -1$ on exactly half (that is, 2^{l-k-1}) of these subsets (in the second equality above) and equals $+1$ on the other half. Thus, on each subset $A_{r,k}$ in the partition for $\triangle_k(n)$, $\triangle_l(n)$ takes value $+1$ on exactly half of the subset and -1 on the other half. Further, $\triangle_k(n)$ has a fixed value on whole of $A_{r,k}$. Thus,

$$\sum_{n=2^{N-1}m}^{2^{N-1}m+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} = 0, k \neq l.$$

Now any integer $s \geq 0$ can be written as $s = 2^{N-1}m + p$, $m \geq 0$, $p \in \{0, \ldots, 2^{N-1}-1\}$. Also for $n = 2^{N-1}m + r$ with $r \in \{0, \ldots, 2^{N-1}-1\}$, $\triangle_k(n) = \triangle_k(r)$ and $\triangle_l(n) = \triangle_l(r)$. Thus,

$$\begin{aligned} \sum_{n=s}^{s+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} &= \sum_{n=2^{N-1}m+p}^{2^{N-1}m+2^{N-1}+p-1} \frac{\triangle_k(n)}{\triangle_l(n)} \\ &= \sum_{n=2^{N-1}m+p}^{2^{N-1}m+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} + \sum_{n=2^{N-1}m+2^{N-1}}^{2^{N-1}m+2^{N-1}+p-1} \frac{\triangle_k(n)}{\triangle_l(n)}. \end{aligned}$$

Now, $\triangle_k(2^{N-1}m + 2^{N-1} + r) = \triangle_k(2^{N-1}m + r) = \triangle_k(r)$, $\forall m \geq 0$. Thus,

$$\begin{aligned} \sum_{n=s}^{s+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} &= \sum_{n=2^{N-1}m+p}^{2^{N-1}m+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} + \sum_{n=2^{N-1}m}^{2^{N-1}m+p-1} \frac{\triangle_k(n)}{\triangle_l(n)} \\ &= \sum_{n=2^{N-1}m}^{2^{N-1}m+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} = 0. \end{aligned}$$

This completes the proof. □

Consider now the algorithm:

$$\theta(m+1) = \pi(\theta(m) + b(m)H(\theta(m), \xi(m))), \tag{15}$$

where $\theta(\cdot) \in \mathcal{R}^N$, $\pi(\cdot)$ as earlier and such that $\| H(\cdot,\cdot) \| \leq \bar{K} < \infty$. The norm $\| \cdot \|$ here and in the rest of this section corresponds to the sup norm. Here, $\xi(m)$ corresponds to noise which could be randomized, deterministic or some combination of both. The following results will be used in the proof of Theorem 2.4.

LEMMA 2.2. *Under* (A3), *for any stochastic approximation algorithm of the form* (15), *given any fixed integer* $P > 0$, $\| \theta(m+k) - \theta(m) \| \to 0$ *as* $m \to \infty$, *for all* $k \in \{1, \ldots, P\}$.

PROOF. Note that (15) can be written as

$$\theta(m+1) = \theta(m) + b(m)H(\theta(m), \xi(m)) + b(m)Z(m), \tag{16}$$

where $Z(m)$ corresponds to the error term because of the projection (see Kushner and Clark [1978]). Thus, for $k \in \{1, \ldots, P\}$, (16) recursively gives

$$\theta(m+k) = \theta(m) + \sum_{j=m}^{m+k-1} b(j)H(\theta(j), \xi(j)) + \sum_{j=m}^{m+k-1} b(j)Z(j).$$

Thus,

$$\| \theta(m+k) - \theta(m) \| \leq \sum_{j=m}^{m+k-1} b(j)(\| H(\theta(j), \xi(j)) \| + \| Z(j) \|) \leq 2\bar{K} \sum_{j=m}^{m+k-1} b(j), \tag{17}$$

since $\| Z(j) \| \leq \bar{K}$ as well. Note from definition of $\{b(j)\}$ that $b(j) \geq b(j+1)$ $\forall j$. Thus,

$$\sum_{j=m}^{m+k-1} b(j) \leq kb(m).$$

Hence, from (17), we have

$$\| \theta(m+k) - \theta(m) \| \leq 2\bar{K}kb(m).$$

The claim now follows from the fact that $b(m) \to 0$ as $m \to \infty$ and $\| \theta(m+k) - \theta(m) \| \geq 0$ $\forall m$. □

Consider now algorithm (12), however, with deterministic perturbation sequences $\{\triangle(n)\}$ as described in Section 2.2. Note that (12) satisfies the conclusions of Lemma 2.2, since terms corresponding to $H(\theta(m), \xi(m))$ in these are uniformly bounded because of (A2) and the fact that C is a compact set. We now have

COROLLARY 2.3. *For* $\{\theta(n)\}$ *defined by* (12) *but with* $\{\triangle(n)\}$ *a deterministic lexicographically ordered sequence, the following holds under* (A2)-(A3) *for any* $m \geq 0$, $k, l \in \{1, \ldots, N\}$, $k \neq l$:

$$\left\| \sum_{n=m}^{m+2^{N-1}-1} \frac{b(n)}{b(m)} \frac{\triangle_k(n)}{\triangle_l(n)} \nabla_k J(\theta(n)) \right\| \to 0 \ \ as \ \ m \to \infty.$$

PROOF. By choosing $P = 2^{N-1}$ in Lemma 2.2, we have $\|\theta(m+s)-\theta(m)\| \to 0$ as $m \to \infty$, for all $s = 1, \ldots, 2^{N-1}$. By (A2), we have $\|\nabla_k J(\theta(m+s)) - \nabla_k J(\theta(m))\| \to 0$ as $m \to \infty$, for all $s = 1, \ldots, 2^{N-1}$. Note that by (A3), for $j \in \{m, m+1, \ldots, m+2^{N-1}-1\}$, $b(j)/b(m) \to 1$ as $m \to \infty$. From Theorem 2.1,

$$\sum_{n=m}^{m+2^{N-1}-1} \frac{\triangle_k(n)}{\triangle_l(n)} = 0 \forall m \geq 0.$$

Thus, one can split any set of type $A_m \stackrel{\triangle}{=} \{m, m+1, \ldots, m+2^{N-1}-1\}$ into two disjoint subsets $A^+_{m,k,l}$ and $A^-_{m,k,l}$ each having the same number of elements, with $A^+_{m,k,l} \cup A^-_{m,k,l} = A_m$ and such that $\triangle_k(n)/\triangle_l(n)$ takes value $+1$ on $A^+_{m,k,l}$ and -1 on $A^-_{m,k,l}$, respectively. Thus,

$$\left\| \sum_{n=m}^{m+2^{N-1}-1} \frac{b(n)}{b(m)} \frac{\triangle_k(n)}{\triangle_l(n)} \nabla_k J(\theta(n)) \right\|$$

$$= \left\| \sum_{n \in A^+_{m,k,l}} \frac{b(n)}{b(m)} \nabla_k J(\theta(n)) - \sum_{n \in A^-_{m,k,l}} \frac{b(n)}{b(m)} \nabla_k J(\theta(n)) \right\|.$$

From these results, we have

$$\left\| \sum_{n=m}^{m+2^{N-1}-1} \frac{b(n)}{b(m)} \frac{\triangle_k(n)}{\triangle_l(n)} \nabla_k J(\theta(n)) \right\| \to 0$$

as $m \to \infty$. □

Consider now the ODE:

$$\dot{\theta}(t) = \tilde{\pi}(-\nabla J(\theta(t))), \tag{18}$$

where for any bounded continuous function $v(\cdot)$,

$$\tilde{\pi}(v(\theta(t))) = \lim_{\eta \downarrow 0} \left(\frac{\pi(\theta(t) + \eta v(\theta(t))) - \theta(t)}{\eta} \right).$$

The operator $\tilde{\pi}(\cdot)$ forces the ODE (18) to evolve within the constraint set C. The fixed points of the ODE lie within the set $K = \{\theta \in C \mid \tilde{\pi}(\nabla J(\theta)) = 0\}$. For the ODE (18), the function $J(\theta)$ itself serves as a Liapunov function since $J(\theta)$ is continuously differentiable by (A2) and

$$\frac{dJ(\theta)}{dt} = \nabla J(\theta) \cdot \dot{\theta} = \nabla J(\theta) \cdot \tilde{\pi}(-\nabla J(\theta)) < 0 \quad \text{on} \quad \{\theta \in C \mid \tilde{\pi}(\nabla J(\theta)) \neq 0\},$$

that is, the trajectories of $J(\theta)$ decrease strictly outside the set K. Thus, by the Liapunov theorem, K contains all asymptotically stable equilibrium points of the ODE (18). In other words, starting from any initial condition, the trajectories of the ODE (18) would converge to a point that lies within the set K. The rest of the analysis works towards approximating the limiting behavior of the

algorithm with that of the corresponding limiting behavior of the trajectories of the ODE (18), that is, shows that the algorithm itself converges to a point within the set K. However, since the algorithm has fixed δ, we show that it converges to a set K' that contains K and such that the 'remainder set' $K' \backslash K$ is "small". In fact, Theorem 2.4 tells us that given that the user decides the amount of the 'remainder set', one can find a $\delta_0 > 0$, such that if one were to use any $\delta \leq \delta_0$, one would be guaranteed convergence to K'. Again, this does not preclude convergence to a fixed point; it only gives a worst-case scenario, that is, in the worst case, the algorithm can converge to a point that lies on the boundary of K'. Since Theorem 2.4 does not specify how to choose the value of δ, the user must select a small enough value of δ for the particular application. In all of our experiments, $\delta = 0.1$ seemed to work well, as similar results were also obtained for $\delta = 0.01$ and $\delta = 0.05$. In the case of algorithms that do not project back iterates on to fixed sets, δ must not be chosen to be too small initially since it could considerably increase the variance of estimates at the start of iterations, which is however not the case with our algorithms. A slowly decreasing δ-sequence (instead of a fixed δ) can also be incorporated in our algorithms.

For a given $\eta > 0$, let $K^\eta \stackrel{\triangle}{=} \{\theta \in C \mid \| \theta - \theta_0 \| < \eta, \theta_0 \in K\}$ be the set of points that are within a distance η from the set K. We then have

THEOREM 2.4. *Under* (A1)–(A3)*, given* $\eta > 0$*, there exists* $\delta_0 > 0$ *such that for all* $\delta \in (0, \delta_0]$*,* $\{\theta(n)\}$ *defined by SPSA1-2L converges to* K^η *almost surely.*

PROOF (SKETCH). Recall from a previous discussion that SPSA1-2L is asymptotically equivalent to (12) under (A1). For notational simplicity, let $\hat{H}_i(m)$ represent

$$\hat{H}_i(m) \stackrel{\triangle}{=} \frac{J(\theta(m) - \delta\triangle(m)) - J(\theta(m) + \delta\triangle(m))}{2\delta\triangle_i(m)}.$$

First, write (12) as

$$\theta_i(m+1) = \theta_i(m) + b(m)\hat{H}_i(m) + b(m)Z_i(m), \quad i = 1, \ldots, N. \tag{19}$$

Here, $Z_i(m)$, $i = 1, \ldots, N$, correspond to the error terms because of the projection operator. Now (19) can be iteratively written as

$$\theta_i(m + 2^{N-1}) = \theta_i(m) + \sum_{j=m}^{m+2^{N-1}-1} b(j)\hat{H}_i(j) + \sum_{j=m}^{m+2^{N-1}-1} b(j)Z_i(j).$$

Using a Taylor series expansions of $J(\theta(m) - \delta\triangle(m))$ and $J(\theta(m) + \delta\triangle(m))$ in $\hat{H}_i(m)$ around the point $\theta(m)$, one obtains by (A2)

$$\begin{aligned}\theta_i(m + 2^{N-1}) \;=\; & \theta_i(m) - \sum_{l=m}^{m+2^{N-1}-1} b(l)\nabla_i J(\theta(l)) \\ & - b(m) \sum_{l=m}^{m+2^{N-1}-1} \sum_{j=1, j\neq i}^{N} \frac{b(l)}{b(m)} \frac{\triangle_j(l)}{\triangle_i(l)} \nabla_j J(\theta(l)) \\ & + b(m)O(\delta) + \sum_{j=m}^{m+2^{N-1}-1} b(j)Z_i(j),\end{aligned} \tag{20}$$

where the $O(\delta)$ corresponds to higher-order terms. Now by Lemma 2.2–Corollary 2.3, under (A2)–(A3) and a repeated application of the triangle inequality

$$\left\| \sum_{l=m}^{m+2^{N-1}-1} \sum_{j=1, j\neq i}^{N} \frac{b(l)}{b(m)} \frac{\triangle_j(l)}{\triangle_i(l)} \nabla_j J(\theta(l)) \right\| \to 0 \text{ as } m \to \infty.$$

The algorithm (12) can therefore be seen to be asymptotically equivalent (cf. Kushner and Clark [1978]) to

$$\theta_i(m+1) = \pi_i(\theta_i(m) - b(m)\nabla_i J(\theta(m))),$$

in the limit as $\delta \to 0$, which in turn can be viewed as the discretization of the ODE (18). The rest can be shown in a similar manner as in Bhatnagar et al. [2001a]. □

Finally, Theorem 2.4 is valid for all algorithms presented in this article and not just SPSA1-2L. For the other algorithms, we mention where needed the required modifications in the proof.

2.4 Algorithms SPSA2-2R and SPSA2-2L

We present first the randomized algorithm SPSA2-2R of Bhatnagar et al. [2001a]. We assume the perturbation sequence $\{\triangle(m)\}$ here to be exactly as in SPSA1-2R.

SPSA2-2R

Let $\{X_l^-\}$ and $\{X_l^+\}$ be the two parallel simulations. These depend on parameter sequences $\{\theta(n) - \delta\triangle(n)\}$ and $\{\theta(n) + \delta\triangle(n)\}$, respectively, as follows: Let $L \geq 1$ be the (integer) observation length over which $\theta(n)$ and $\triangle(n)$ are held fixed between updates of $\theta(n)$ according to the recursive update below. In other words, for $n \geq 0$ and $m \in \{0, \ldots, L-1\}$, X_{nL+m}^- and X_{nL+m}^+ are governed by $\theta(n) - \delta\triangle(n)$ and $\theta(n) + \delta\triangle(n)$, respectively. We also define sequences $\{Z^-(l)\}$ and $\{Z^+(l)\}$ for averaging the cost function as follows: $Z^-(0) = Z^+(0) = 0$ and for $n \geq 0$, $m \in \{0, \ldots, L-1\}$,

$$Z^-(nL+m+1) = Z^-(nL+m) + b(n)(h(X_{nL+m}^-) - Z^-(nL+m)), \tag{21}$$

$$Z^+(nL+m+1) = Z^+(nL+m) + b(n)(h(X_{nL+m}^+) - Z^+(nL+m)). \tag{22}$$

The parameter update recursion is then given by

$$\theta_i(n+1) = \pi_i\left(\theta_i(n) + a(n)\left[\frac{Z^-(nL) - Z^+(nL)}{2\delta\triangle_i(n)}\right]\right). \tag{23}$$

This algorithm uses coupled stochastic recursions that are individually driven by different timescales. Thus, when viewed from the faster timescale (i.e., recursions (21)–(22)), the slower recursion (23) appears to be quasi-static, while from the slower timescale, the faster recursions appear to be essentially equilibrated.

The analysis for convergence proceeds as for SPSA1-2R, with the slower timescale recursion (23) being asymptotically equivalent under (A1) to (12) but with step-size schedules $a(m)$ in place of $b(m)$. Using analogous Taylor series arguments and approximation analysis, the algorithm can be seen to asymptotically track the stable points of the ODE (18).

SPSA2-2L

This algorithm is exactly the same as SPSA2-2R but with a deterministic perturbation sequence $\{\triangle(m)\}$ obtained using the lexicographical ordering described in Section 2.2. The convergence analysis for this algorithm works in exactly the same manner as for SPSA1-2L, but with step-size schedules $a(m)$ in place of $b(m)$ and which also satisfy the desired properties.

Next we present the one-simulation randomized difference versions of types 1 and 2 algorithms.

2.5 One-Simulation Algorithms SPSA1-1R and SPSA2-1R

Let $\{\triangle(m)\}$ be a randomized difference vector sequence as in SPSA1-2R and SPSA2-2R, respectively. We have the following algorithms.

SPSA1-1R

Consider the process $\{\hat{X}_j\}$ governed by $\{\hat{\theta}_j\}$ with $\hat{\theta}_j = \theta(m) + \delta\triangle(m)$ for $n_m < j \leq n_{m+1}$, $m \geq 0$. Then, for $i = 1, \ldots, N$,

$$\theta_i(m+1) = \pi_i\left(\theta_i(m) - \sum_{j=n_m+1}^{n_{m+1}} a(j)\left(\frac{h(\hat{X}_j)}{\delta\triangle_i(m)}\right)\right). \tag{24}$$

SPSA2-1R

Consider the process $\{\tilde{X}_l\}$ governed by $\{\tilde{\theta}_l\}$ defined by $\tilde{\theta}_l = \theta(n) + \delta\triangle(n)$ for $n = [l/L]$, where $L \geq 1$ is a given fixed integer as in SPSA2-2R and $[l/L]$ represents the integer part of l/L. Defining sequence $\{\tilde{Z}(l)\}$ in the same manner as $\{Z^+(l)\}$ in SPSA2-2R with $\tilde{Z}(0) = 0$, we have

$$\theta_i(n+1) = \pi_i\left(\theta_i(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_i(n)}\right)\right), \tag{25}$$

where for $m = 0, 1, \ldots, L-1$,

$$\tilde{Z}(nL+m+1) = \tilde{Z}(nL+m) + b(n)(h(\tilde{X}_{nL+m}) - \tilde{Z}(nL+m)). \tag{26}$$

Both algorithms SPSA1-1R and SPSA2-1R use only one simulation as opposed to two required in SPSA1-2R and SPSA2-2R. As in the case of SPSA1-2R and SPSA2-2R, one can show as in Bhatnagar et al. [2001a] that SPSA1-1R and iteration (25) of SPSA2-1R are asymptotically equivalent under (A1) to the following algorithm:

$$\theta_i(m+1) = \pi_i\left(\theta_i(m) - c(m)\left(\frac{J(\theta(m) + \delta\triangle(m))}{\delta\triangle_i(m)}\right)\right), \tag{27}$$

where $c(m) = b(m)$ in SPSA1-1R ($a(m)$ in SPSA2-1R). By a Taylor series expansion around the point $\theta(m)$, one obtains under (A2)

$$J(\theta(m) + \delta\triangle(m)) = J(\theta(m)) + \delta \sum_{j=1}^{N} \triangle_j(m)\nabla_j J(\theta(m)) + O(\delta^2).$$

Thus,

$$\frac{J(\theta(m) + \delta\triangle(m))}{\delta\triangle_i(m)} = \frac{J(\theta(m))}{\delta\triangle_i(m)} + \nabla_i J(\theta(m)) + \sum_{j=1, j\neq i}^{N} \frac{\triangle_j(m)}{\triangle_i(m)}\nabla_j J(\theta(m)) + O(\delta). \tag{28}$$

The bias $\hat{b}_m^\delta(\theta(m), \triangle(m))$ in the gradient estimate in (27) is

$$\hat{b}_m^\delta(\theta(m), \triangle(m)) = E\left[\left(\frac{J(\theta(m) + \delta\triangle(m))}{\delta\triangle_i(m)} - \nabla_i J(\theta(m))\right) \;\middle|\; \theta(m)\right].$$

From (28) and from the definition of $\{\triangle(m)\}$,

$$\hat{b}_m^\delta(\theta(m), \triangle(m)) = \frac{J(\theta(m))}{\delta} E\left[\frac{1}{\triangle_i(m)}\right] + \sum_{j=1, j\neq i}^{N} E\left[\frac{\triangle_j(m)}{\triangle_i(m)}\right] \nabla_j J(\theta(m)) + O(\delta). \tag{29}$$

From the definition of $\{\triangle(m)\}$, it is easy to see that $E[1/\triangle_i(m)] = 0$ and $E[\triangle_j(m)/\triangle_i(m)] = 0$, for $j \neq i$. These quantities also equal zero in the case of perturbations with distributions that satisfy (the more general) Condition (B). Thus, $\hat{b}_m^\delta(\theta(m), \triangle(m)) = O(\delta)$. The gradient estimate in the two-simulation randomized difference form (12) also gives an $O(\delta)$ bias. However, an analogous Taylor series expansion of the estimate in (12) results in a direct cancellation of the terms containing $J(\theta(m))$, instead of the same being mean-zero in (29). Next we describe the deterministic versions of the one-simulation algorithms with lexicographical ordering.

2.6 Algorithms SPSA1-1L and SPSA2-1L

There is a small change in the lexicographic construction for algorithms SPSA1-1L and SPSA2-1L, respectively, which we first describe. The bias in the gradient estimate in (27) under a deterministic $\{\triangle(m)\}$ sequence is given by

$$\left(\frac{J(\theta(m) + \delta\triangle(m))}{\delta\triangle_i(m)} - \nabla_i J(\theta(m))\right) = \frac{J(\theta(m))}{\delta\triangle_i(m)} + \sum_{j=1, j\neq i}^{N} \frac{\triangle_j(m)}{\triangle_i(m)}\nabla_j J(\theta(m)) + O(\delta). \tag{30}$$

Thus, for the one-simulation algorithms, the terms contributing to the bias are both the perturbation ratios ($\triangle_j(m)/\triangle_i(m)$) and inverses ($1/\triangle_i(m)$), as opposed to just perturbation ratios in the two-simulation algorithms. Thus, the lexicographic construction needs to be altered, as we can no longer hold $\triangle_1(m) = -1\ \forall m$, since that would contribute to the bias. We therefore need

a construction for which both perturbation ratios and inverses become zero over cycles. This is trivially achieved by moving the perturbation sequence cyclically over the full space E of perturbations. We again assume E is ordered lexicographically.

SPSA1-1L

This algorithm is exactly the same as SPSA1-1R but with a deterministic lexicographically ordered perturbation sequence as described above.

SPSA2-1L

This algorithm is exactly the same as SPSA2-1R but with a deterministic lexicographic perturbation sequence as described above.

For the lexicographic construction described above (for the one-simulation algorithms), we have the following basic result:

THEOREM 2.5. *The lexicographic perturbations* $\triangle(n) \stackrel{\triangle}{=} (\triangle_1(n), \ldots, \triangle_N(n))^T$ *for one-simulation algorithms satisfy:*

$$\sum_{n=s}^{s+2^N-1} \frac{1}{\triangle_k(n)} = 0 \text{ and } \sum_{n=s}^{s+2^N-1} \frac{\triangle_i(n)}{\triangle_j(n)} = 0$$

for any $s \geq 0$, $i \neq j$, $i, j, k \in \{1, \ldots, N\}$.

PROOF. Follows in a similar manner as the proof of Theorem 2.1. □

Consider now the algorithm (15) with step-size $b(m)$ (respectively, $a(m)$) for SPSA1-1L (respectively, SPSA2-1L). The conclusions of Lemma 2.2 continue to hold under (A3) even with step-size $a(m)$. We also have:

COROLLARY 2.6. *For* $\{\theta(n)\}$ *defined by* (27) *but with* $\{\triangle(n)\}$ *a deterministic lexicographic sequence for algorithms SPSA1-1L and SPSA2-1L, respectively, the following holds under* (A2)–(A3) *for any* $m \geq 0$, $i, k, l \in \{1, \ldots, N\}$, $k \neq l$:

$$\left\| \sum_{j=m}^{m+2^N-1} \frac{c(j)}{c(m)} \frac{1}{\triangle_i(j)} J(\theta(j)) \right\|, \left\| \sum_{j=m}^{m+2^N-1} \frac{c(j)}{c(m)} \frac{\triangle_k(j)}{\triangle_l(j)} \nabla_k J(\theta(j)) \right\| \to 0 \text{ as } m \to \infty,$$

where $c(l) = b(l)$ *(respectively,* $a(l)$*)* $\forall l$ *for SPSA1-1L (respectively, SPSA2-1L).*

PROOF. Follows in a similar manner as that of Corollary 2.3, using conclusions of Theorem 2.5 and choosing $P = 2^N$ in the conclusions of Lemma 2.2. □

Finally, the conclusions of Theorem 2.4 can similarly be shown to hold under (A1)–(A3) for algorithms SPSA1-1L and SPSA2-1L as well. In the next section, we give a second construction of deterministic perturbation sequences based on Hadamard matrices.

3. A CONSTRUCTION OF DETERMINISTIC PERTURBATIONS BASED ON HADAMARD MATRICES

As illustrated in the previous section (particularly Theorem 2.1 and Corollary 2.3), the essential properties of the deterministic perturbation

sequence $\{\Delta(n)\}$ for convergence of SPSA algorithms are the following:

(P.1) There exists a $P \in \mathcal{N}$ such that for every $i, j \in \{1, \ldots, N\}$, $i \neq j$ and for any $s \in \mathcal{N}$,

$$\sum_{n=s}^{s+P} \frac{\Delta_i(n)}{\Delta_j(n)} = 0. \tag{31}$$

(P.2) There exists a $P \in \mathcal{N}$ such that for every $k \in \{1, \ldots, N\}$ and any $s \in \mathcal{N}$,

$$\sum_{n=s}^{s+P} \frac{1}{\Delta_k(n)} = 0. \tag{32}$$

Property (P.1) is required for convergence of the two-simulation SPSA algorithms, while both properties (P.1–2) are required for convergence of the one-simulation version of these algorithms. In this section, we present an alternative way of constructing deterministic periodic perturbations satisfying (P.1–2) that can lead to significantly smaller period P than the lexicographic construction introduced in the previous sections. It suffices to construct a set of P distinct vectors, $\{\Delta(1), \ldots, \Delta(P)\}$, in $\{\pm 1\}^N$ that satisfy properties (P.1–2) with the summations taken over the entire set of vectors. The desired perturbation sequence can then be obtained by repeating the set of constructed perturbations in the same arbitrary order.

We first show that construction of P vectors in $\{\pm 1\}^N$ satisfying (P.1–2) is equivalent to construction of N mutually orthogonal vectors in $\{\pm 1\}^P$. The first part of the following lemma is based on the result presented in Sandilya and Kulkarni [1997].

LEMMA 3.1. *Let $\Delta(1), \ldots, \Delta(P)$ be P distinct vectors in $\{\pm 1\}^N$. Define N vectors in $\{\pm 1\}^P$, $h(1), \ldots, h(N)$ by*

$$h(n) = [\Delta_n(1), \Delta_n(2), \ldots, \Delta_n(P)]^T,$$

for $n = 1, \ldots, N$. Then the following hold:

—*$\{\Delta(1), \ldots, \Delta(P)\}$ satisfy property* (P.1) *if and only if $h(1), \ldots, h(N)$ are mutually orthogonal.*

—*$\{\Delta(1), \ldots, \Delta(P)\}$ satisfy property* (P.2) *if and only if $\sum_{n=1}^{P} h_n(k) = 0$ for all $k = 1, \ldots, N$. That is, each vector $h(k)$ has exactly half $(P/2)$ of its elements being 1's.*

PROOF. The proof is straightforward. □

From Lemma 3.1, the following lower bound on P can be established:

COROLLARY 3.2. *For any $\{\Delta(1), \ldots, \Delta(P)\}$ satisfying* (P.1)*, we have $P \geq 2\lceil \frac{N}{2} \rceil$.*

PROOF. Note that P is always even, since otherwise the inner product of any two vectors (with elements in $\{1, -1\}$) in $\{\pm 1\}^P$ cannot be zero. Hence, it suffices to show that $P \geq N$. We prove by contradiction.

Assume that there exists a set of P' distinct vectors $\{\Delta(1), \ldots, \Delta(P')\}$ satisfying (P.1) with $P' < N$. Then, by Lemma 3.1, we have N orthogonal vectors

$h(1), \ldots, h(N)$ (as defined in Lemma 3.1) in $\{\pm 1\}^{P'}$. This is certainly not possible with $P' < N$. Therefore, we have $P \geq N$. □

Remark. Even though we consider perturbation vectors in spaces $\{\pm 1\}^N$ and $\{\pm 1\}^P$, some of our results (for instance Lemma 3.1) are more general and can be applied to $\mathcal{R}^N$ and $\mathcal{R}^P$, respectively.

Based on Lemma 3.1, we propose a class of deterministic perturbations constructed from *Hadamard* [Hadamard 1893; Seberry and Yamada 1992] matrices with appropriate dimensions. An $m \times m$ ($m \geq 2$) matrix H is called a Hadamard matrix of order m if its entries belong to $\{1, -1\}$ and $H^T H = mI_m$, where I_m denotes the $m \times m$ identity matrix. It can be easily seen that all the columns (rows) of a Hadamard matrix are orthogonal. We first show that a set of vectors in $\{\pm 1\}^N$ satisfying (P.1) can be constructed from a Hadamard matrix of order $P \geq N$. The following lemma presents such a construction.

LEMMA 3.3 (CONSTRUCTION FOR TWO-SIMULATION ALGORITHMS). *Let H_P be a Hadamard matrix of order P with $P \geq N$. Let $h(1), \ldots, h(N)$ be any N columns from H_P. Define P distinct vectors $\Delta(1), \ldots, \Delta(P)$ in $\{\pm 1\}^N$ by*

$$\Delta(k) = [h_k(1), \ldots, h_k(N)]^T,$$

for $k = 1, \ldots, P$. Then $\{\Delta(1), \ldots, \Delta(P)\}$ satisfies property (P.1).

PROOF. Follows directly from Lemma 3.1 and the definition of Hadamard matrices. □

Lemma 3.3 basically says the following: To construct the desired deterministic perturbation $\{\Delta(1), \ldots, \Delta(P)\}$, we first select any N columns from a Hadamard matrix of order $P \geq N$ (they are mutually orthogonal), then take the kth element of these vectors (ordered) to form $\Delta(k)$.

We next propose a construction of perturbations that satisfies both properties (P.1–2) for one-simulation algorithms from Hadamard matrices. We first give a simple result for normalized Hadamard matrices. A Hadamard matrix is said to be *normalized* if all the elements of its first column and row are 1's. The following simple lemma enables us to construct perturbations that satisfy both properties (P.1–2) from Hadamard matrices.

LEMMA 3.4. *If H_m is a normalized Hadamard matrix of order m, then every column other than the first column of H_m has exactly $m/2$ 1's, that is, the sum of all the entries in any column other than the first one is zero.*

Based on Lemma 3.4, the next lemma gives a construction of perturbations from a normalized Hadamard matrix.

LEMMA 3.5 (CONSTRUCTION FOR ONE-SIMULATION ALGORITHMS). *Let H_P be a normalized Hadamard matrix of order P with $P \geq N+1$. Let $h(1), \ldots, h(N)$ be any N columns other than the first column from H_P. Define P distinct vectors $\Delta(1), \ldots, \Delta(P)$ in $\{\pm 1\}^N$ by*

$$\Delta(k) = [h_k(1), \ldots, h_k(N)]^T,$$

for $k = 1, \ldots, P$. Then, $\{\Delta(1), \ldots, \Delta(P)\}$ satisfies properties (P.1–2).

Table II. Perturbation Ratios for $N = 4$ for Two-Simulation Algorithms with Hadamard Construction

$\Delta(n)$	$\frac{\Delta_1(n)}{\Delta_2(n)}$	$\frac{\Delta_1(n)}{\Delta_3(n)}$	$\frac{\Delta_1(n)}{\Delta_4(n)}$	$\frac{\Delta_2(n)}{\Delta_3(n)}$	$\frac{\Delta_2(n)}{\Delta_4(n)}$	$\frac{\Delta_3(n)}{\Delta_4(n)}$
$\Delta(1) = [+1, +1, +1, +1]^T$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$
$\Delta(2) = [+1, -1, +1, -1]^T$	-1	$+1$	-1	-1	$+1$	-1
$\Delta(3) = [+1, +1, -1, -1]^T$	$+1$	-1	-1	-1	-1	$+1$
$\Delta(4) = [+1, -1, -1, +1]^T$	-1	-1	$+1$	$+1$	-1	-1

PROOF. Follows directly from Lemma 3.3 and Lemma 3.4. □

Existence of Hadamard matrices of general order has been extensively studied in the area of combinatorial design (Hadamard's original conjecture in Hadamard [1893] that a Hadamard matrix of order m exists for any $m = 4q, q \in \mathcal{N}$, is still an open problem); see, for example, Seberry and Yamada [1992]. It has been shown that a Hadamard matrix of order m exists for $m = 4q$ for most $q < 300$ [Seberry and Yamada 1992]. Clearly, based on the proposed constructions, such a matrix of smaller order can lead to periodic perturbations with smaller period. Here, we present a simple and systematic way of constructing normalized Hadamard matrices of order $m = 2^k$, $k \in \mathcal{N}$. Our construction is sequential in k:

—For $k = 1$, let

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

—For general $k > 1$,

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}.$$

It can be easily checked that this sequence of matrices generates normalized Hadamard matrices. Following the constructions presented in Lemma 3.3 and Lemma 3.5, we can construct periodic perturbations with period $P = 2^{\lceil log_2 N \rceil}$ for two-simulation algorithms, and perturbations with period $P = 2^{\lceil log_2 (N+1) \rceil}$ for one-simulation algorithms.

Here, we give examples for $N = 4$. Following Lemma 3.3, we construct $\Delta(1), \ldots, \Delta(4)$ for two-simulation algorithms from H_4: (we basically take the row vectors as the perturbations)

$$\begin{aligned} \Delta(1) &= [1, 1, 1, 1]^T, \\ \Delta(2) &= [1, -1, 1, -1]^T, \\ \Delta(3) &= [1, 1, -1, -1]^T, \\ \Delta(4) &= [1, -1, -1, 1]^T. \end{aligned}$$

It can be easily check that these perturbations have the desired property (P.1) (cf. Table II). To construct perturbations for one-simulation algorithms, we start

Table III. Perturbation Ratios and Inverses for $N = 4$ for One-Simulation Algorithms using Hadamard Construction

$\frac{\Delta_1(n)}{\Delta_2(n)}$	$\frac{\Delta_1(n)}{\Delta_3(n)}$	$\frac{\Delta_1(n)}{\Delta_4(n)}$	$\frac{\Delta_2(n)}{\Delta_3(n)}$	$\frac{\Delta_2(n)}{\Delta_4(n)}$	$\frac{\Delta_3(n)}{\Delta_4(n)}$	$\frac{1}{\Delta_1(n)}$	$\frac{1}{\Delta_2(n)}$	$\frac{1}{\Delta_3(n)}$	$\frac{1}{\Delta_4(n)}$
$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$
-1	$+1$	-1	-1	$+1$	-1	-1	$+1$	-1	$+1$
-1	-1	$+1$	$+1$	-1	-1	$+1$	-1	-1	$+1$
$+1$	-1	-1	-1	-1	$+1$	-1	-1	$+1$	$+1$
$+1$	$+1$	-1	$+1$	-1	-1	$+1$	$+1$	$+1$	-1
-1	$+1$	$+1$	-1	-1	$+1$	-1	$+1$	-1	-1
-1	-1	-1	$+1$	$+1$	$+1$	$+1$	-1	-1	-1
$+1$	-1	$+1$	-1	$+1$	-1	-1	-1	$+1$	-1

from a normalized Hadamard matrix H_8.

$$H_8 = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right].$$

We then take columns 2–5 (or any four columns except the first one) of H_8 to construct the desired perturbations. So we have (by taking the rows of columns 2–5 from H_8)

$$\begin{aligned} \Delta(1) &= [1, 1, 1, 1]^T, \\ \Delta(2) &= [-1, 1, -1, 1]^T, \\ \Delta(3) &= [1, -1, -1, 1]^T, \\ \Delta(4) &= [-1, -1, 1, 1]^T, \\ \Delta(5) &= [1, 1, 1, -1]^T, \\ \Delta(6) &= [-1, 1, -1, -1]^T, \\ \Delta(7) &= [1, -1, -1, -1]^T, \\ \Delta(8) &= [-1, -1, 1, -1]^T. \end{aligned}$$

Note that the same matrix (H_8) would also work for $N = 5, 6$ and 7 as well. It can be checked that the constructed perturbations satisfy properties (P.1–2) (cf. Table III).

3.1 Algorithms SPSA1-2H, SPSA2-2H, SPSA1-1H and SPSA2-1H

The two-simulation algorithms SPSA1-2H and SPSA2-2H are similar to SPSA1-2R and SPSA2-2R, respectively, but with deterministic perturbation sequences $\{\triangle(m)\}$ constructed using normalized Hadamard matrices as in Lemma 3.3. The one-simulation algorithms SPSA1-1H and SPSA2-1H, on the other hand, are analogous to their one-simulation randomized difference counterparts SPSA1-1R and SPSA2-1R, respectively, but with deterministic

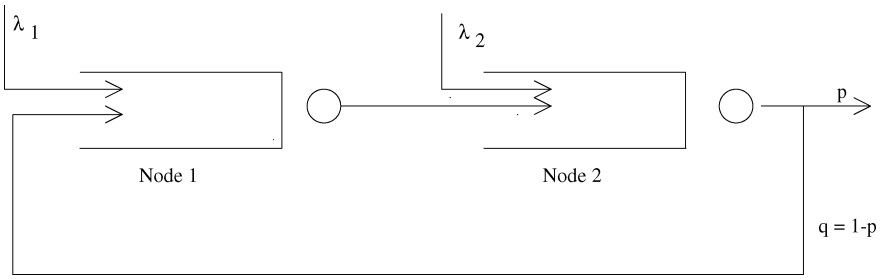

Fig. 1. Queueing network.

perturbation sequences $\{\triangle(m)\}$ obtained using the normalized Hadamard matrices as explained in Lemma 3.5.

4. NUMERICAL RESULTS

We show numerical experiments on a two-node network of $M/G/1$ queues (shown in Figure 1), with all algorithms SPSA1-2L, SPSA1-2H, SPSA2-2L, SPSA2-2H, SPSA1-1R, SPSA2-1R, SPSA1-1L, SPSA1-1H, SPSA2-1L and SPSA2-1H proposed in this article and provide comparisons with algorithms SPSA1-2R and SPSA2-2R of Bhatnagar et al. [2001a].

Nodes 1 and 2 in the network are fed with independent external arrival streams with respective rates $\lambda_1 = 0.2$ and $\lambda_2 = 0.1$. The departures from Node 1 enter Node 2. Further, departures from Node 2 are fed back with probability $q = 0.6$ to the first node. The service time processes $\{S_n^i(\theta^i)\}$ at the two nodes $i = 1, 2$, are defined by $S_n^i(\theta^i) = U_n^i(1 + \prod_{j=1}^{M} |\theta_j^i(n) - \bar{\theta}_j^i|)/R_i$, $i = 1, 2$, $n \geq 1$, where $U_n^i \sim U(0, 1)$ and $R_1 = 10$ and $R_2 = 20$. Also $\theta_1^i(n), \ldots, \theta_M^i(n)$ represent the nth update of the parameter components of service time at Node i, and $\bar{\theta}_1^i, \ldots, \bar{\theta}_M^i$ represent the target parameter components. Note that if U_n^i is replaced by $-\log(U_n^i)$, $S_n^i(\theta^i)$ would be exponentially distributed (by the inverse transform technique for random variate generation, see Law and Kelton [2000]) with rate $R_i/(1 + \prod_{j=1}^{M} |\theta_j^i(n) - \bar{\theta}_j^i|)$, which is the setting considered in Bhatnagar et al. [2001a]. We assume each $\theta_j^i(n)$ is constrained according to $0.1 \leq \theta_j^i(n) \leq 0.6$, $j = 1, \ldots, M, i = 1, 2, \forall n$. We set $\bar{\theta}_j^i = 0.3$ for all $i = 1, 2, j = 1, \ldots, M$. The initial $\theta_j^1(0) = 0.4$, $j = 1, \ldots, M$, and $\theta_j^2(0) = 0.2$, $j = 1, \ldots, M$. As in Bhatnagar et al. [2001a, 2001b], the step-size sequences $\{a(n)\}$ and $\{b(n)\}$ are defined according to (8), with $\hat{a} = \hat{b} = 1$, $\alpha = 2/3$. Moreover, we choose $L = 100$ in all algorithms of type 2. We used $\delta = 0.1$ for all algorithms.

The cost function is chosen to be the sum of waiting times of individual customers at the two nodes. Suppose W_n^i is the waiting time of the nth arriving customer at ith node and r_n^i is the residual service time of the customer in service at the instant of nth arrival at ith node. Then $\{(W_n^1, r_n^1, W_n^2, r_n^2)\}$ is Markov. Thus, for our experiments, we assume cost function $h(W_n^1, r_n^1, W_n^2, r_n^2) = W_n^1 + W_n^2$. Note that here the portions r_n^1, r_n^2 of the state are not observed or stay *hidden*. It is quite often the case with many practical applications that certain portions of the state are not observed. Our algorithms work in the case of such

applications as well even though we do not try to estimate in any manner the unobserved portions of the state. Both (A1) and (A2) are satisfied by our example in this section. Observe that service times $S_n^i(\theta^i)$ at node i, are bounded from above by the quantity $(1 + 2^M \prod_{j=1}^{M} \theta^i_{j,\max})/R_i$, $i = 1, 2$. In fact, for the chosen values of the various parameters, the service times are bounded from above by $\left(1 + (0.3)^M\right)/R_i$ at nodes $i = 1, 2$. It is easy to see that the average overall interarrival time at either node is greater than the corresponding mean service time at that node (in fact it is greater than even the largest possible service time at either node). Hence, the system remains stable for any given θ, thus satisfying (A1). For (A2) to hold, one needs to show that derivatives of steady state mean waiting time for any fixed θ exist and are continuous, and that their second derivatives also exist. One can show this using sample path arguments and an application of the dominated convergence theorem using similar techniques as in Chapter 4 of Bhatnagar [1997].

We consider the cases $M = 2$, 6 and 15. Thus, the parameter θ corresponds to 4, 12 and 30 dimensional vectors. We define "Distance from Optimum" as the performance measure

$$d(\theta(n), \bar{\theta}) \stackrel{\triangle}{=} \left(\sum_{i=1}^{2} \sum_{j=1}^{M} \left(\theta_j^i(n) - \bar{\theta}_j^i\right)^2 \right)^{1/2}.$$

For the cost to be minimized, one expects $\theta_j^i(n)$ to converge to $\bar{\theta}_j^i$, $j = 1, \ldots, M$, $i = 1, 2$, as $n \to \infty$. In other words, the total average delay of a customer is minimized at the target parameter value $\bar{\theta}$. One expects the algorithms to converge to either $\bar{\theta}$ or a point close to it (since δ is a fixed nonzero quantity), see the discussion before Theorem 2.4. Thus, algorithms that have a lower $\lim_{n\to\infty} d(\theta(n), \bar{\theta})$ value, or which converge faster, would show better performance. In actual simulations though, one needs to terminate the simulation after a certain fixed number of instants. We terminate all simulation runs (for all algorithms) after 6×10^5 estimates of the cost function. In Figures 2 to 5, we illustrate a few sample performance plots for the various algorithms. In Figures 2 and 3, we plot the trajectories of $d(\theta(n), \bar{\theta})$ for one and two simulation, type 1 algorithms for $N = 4$, while in Figures 4 and 5, we plot the same for one and two simulation type 2 algorithms corresponding to $N = 30$. We plot these trajectories for all algorithms by averaging over five independent simulation runs with different initial seeds. In these figures, the trajectory $i - j$K ($i = 1, 2$, K = R, H) denotes the trajectory for the algorithm SPSA$i - j$K. The mean and standard error from all simulations that we performed for all algorithms, for the three cases viz., $N = 4$, 12 and 30, at the termination of the simulations are given in Table IV.

Note that in the case of algorithms that use normalized Hadamard matrix based perturbations, the Hadamard matrices H_P for the parameter dimensions $N = 4$, 12 and 30 are of orders $P = 4$, 16 and 32 for the two-simulation algorithms, while these are of orders $P = 8$, 16 and 32, respectively, for the one-simulation case. For $N = 12$ and 30, we use columns 2–13 and 2–31, respectively, in the corresponding Hadamard matrices, for both one and two-simulation algorithms. For $N = 4$, for the one-simulation case, we use columns

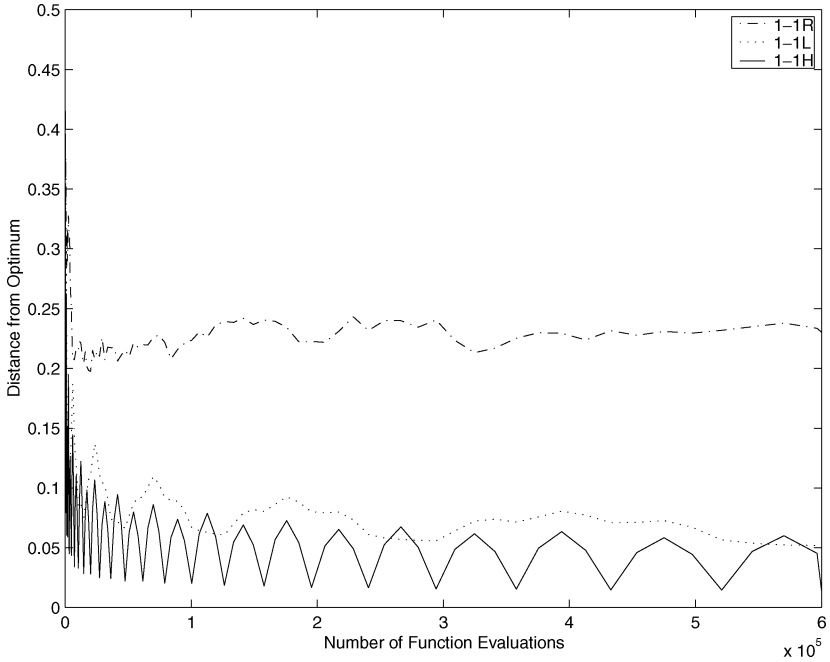

Fig. 2. Convergence of "Distance from Optimum" for one-simulation Type 1 algorithms for $N = 4$.

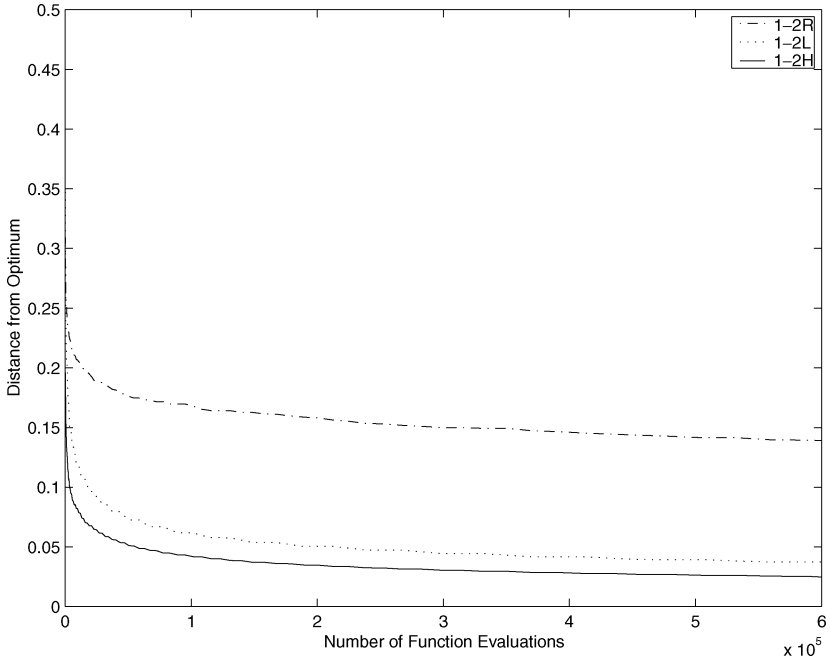

Fig. 3. Convergence of "Distance from Optimum" for two-simulation Type 1 algorithms for $N = 4$.

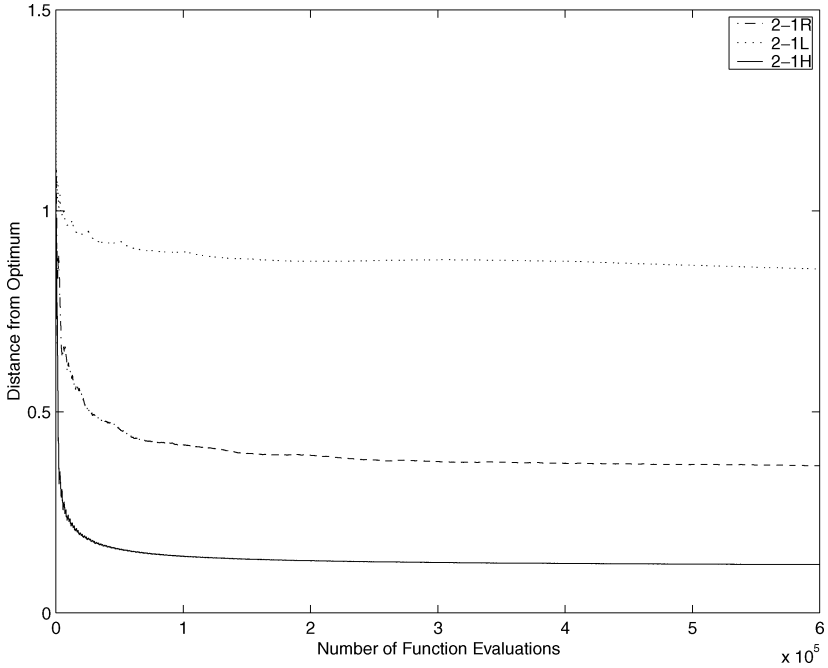

Fig. 4. Convergence of "Distance from Optimum" for one-simulation Type 2 algorithms for $N = 30$.

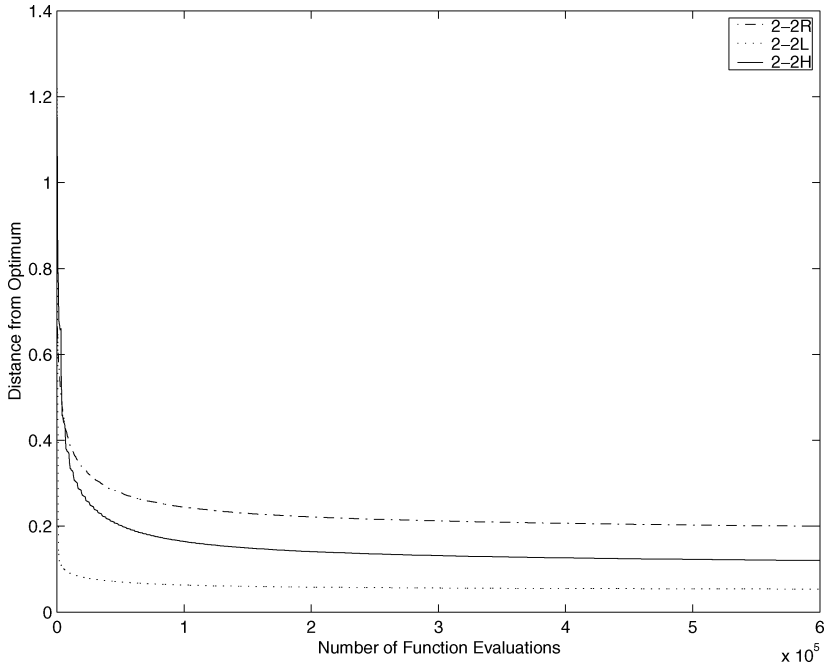

Fig. 5. Convergence of "Distance from Optimum" for two-simulation Type 2 algorithms for $N = 30$.

Table IV. Performance after 6×10^5 Function Evaluations

Algorithm	$d(\theta(n), \bar{\theta})$ $N = 4$	$d(\theta(n), \bar{\theta})$ $N = 12$	$d(\theta(n), \bar{\theta})$ $N = 30$
SPSA1-1R	0.230 ± 0.032	0.404 ± 0.019	0.774 ± 0.017
SPSA1-1L	0.052 ± 0.002	$0.568 \pm 3.4 \times 10^{-4}$	$1.013 \pm 3 \times 10^{-4}$
SPSA1-1H	0.014 ± 0.008	$0.192 \pm 5.8 \times 10^{-4}$	0.292 ± 0.004
SPSA2-1R	0.131 ± 0.030	0.197 ± 0.038	0.366 ± 0.019
SPSA2-1L	0.053 ± 0.024	$0.137 \pm 8.2 \times 10^{-4}$	0.855 ± 0.001
SPSA2-1H	0.033 ± 0.010	0.032 ± 0.011	0.120 ± 0.037
SPSA1-2R	0.139 ± 0.040	0.019 ± 0.003	0.267 ± 0.033
SPSA1-2L	0.037 ± 0.010	0.012 ± 0.003	0.085 ± 0.086
SPSA1-2H	0.025 ± 0.018	0.019 ± 0.004	0.150 ± 0.033
SPSA2-2R	0.022 ± 0.003	0.035 ± 0.009	0.200 ± 0.032
SPSA2-2L	0.013 ± 0.002	0.018 ± 0.027	0.054 ± 0.038
SPSA2-2H	0.011 ± 0.004	0.040 ± 0.028	0.120 ± 0.052

2–5 of the corresponding Hadamard matrix, while for the two-simulation case, all columns of the 4-dimensional matrix are used.

4.1 Discussion on the Performance of Algorithms

Our experiments indicate that deterministic perturbation algorithms perform significantly better than their randomized difference counterparts in most cases. We conjecture that one of the reasons is that a properly chosen deterministic sequence of perturbations will retain the requisite averaging property that leads to the convergence of SPSA, but with lower variance than randomly generated perturbation sequences. However, deterministic sequences introduce additional bias, since the averaging property only holds when the sequence completes a full cycle (as opposed to in expectation for the random sequences), but this bias vanishes asymptotically. Moreover, it is not clear if the quality of the random number generator also plays a role in the case of random perturbation sequences. For our experiments, we used the well-known prime modulus multiplicative linear congruential generator $X_{k+1} = aX_k \bmod m$ with multiplier $a = 7^5 = 16805$ and modulus $m = 2^{31} - 1 = 2147483647$ (cf. Park and Miller [1988] and Law and Kelton [2000]).

Worth noting are the impressive (more than an order of magnitude) performance improvements using the one-simulation deterministic perturbation algorithms SPSA1-1H and SPSA2-1H that use normalized Hadamard construction. Recall from the analysis in Section 3 that one-simulation algorithms have "additional" terms of the type $J(\theta(j))/\delta\triangle_i(j)$ that are likely to contribute heavily towards the bias mainly through the "small" δ term in the denominator. Thus, if $\triangle_i(j)$ does not change sign often enough, the performance of the algorithm is expected to deteriorate. This is indeed observed to be the case with deterministic lexicographic sequences (first construction). Thus, even though SPSA1-1L and SPSA2-1L show better performance than their randomized counterparts when $N = 4$, their performance deteriorates for $N = 30$. On the other hand, the construction based on normalized Hadamard matrices requires far fewer points, with all component $\triangle_i(j)$'s, $i = 1, \ldots, N$, changing sign much more frequently.

In the case of two-simulation algorithms, both the lexicographic and Hadamard constructions show significantly better performance than randomized difference algorithms. SPSA1-2H and SPSA2-2H show the best performance when $N = 4$. However, surprisingly, for both $N = 12$ and $N = 30$, SPSA1-2L and SPSA2-2L show the best performance (which is slightly better than SPSA1-2H and SPSA2-2H as well). Note that for $N = 30$ in our experiments, the perturbation sequence in the lexicographic construction needs to visit $2^{29} \approx 1$ billion points (which is significantly larger than even the total perturbations generated during the full course of the simulation), in order to complete a cycle once, whereas in the Hadamard construction, this number is just 32. Our guess is that while the lexicographic construction converges slower in comparison to the perfect gradient descent (with the actual gradient), it leads to a search pattern that has lower dispersion and hence could explore the space better when the algorithm is away from the optimum.

In general, the performance of the two-simulation SPSA algorithms is better than the corresponding one-simulation algorithms. A similar observation has been made in the case of one-simulation, one-timescale algorithms in Spall [1997] and Spall and Cristion [1998]. However, in our experiments, in most cases, the one-simulation algorithms with SPSA1-1H and SPSA2-1H are better than their respective two-simulation randomized difference counterparts SPSA1-2R and SPSA2-2R, respectively. Therefore, our experiments suggest that for one-simulation algorithms, it is preferable to use the normalized Hadamard matrix-based deterministic perturbation construction. Also, for two-simulation algorithms, for high dimensional parameters, deterministic perturbation sequences with the lexicographic (first) construction seem to be preferable. We do not as yet have a thorough explanation for the improvement in performance in the case of lexicographic sequences, with the discussion above offering a possible reason. Moreover, other constructions with possibly more vectors than the Hadamard matrix-based construction should be tried before conclusive inferences can be drawn. Finally, as noted in Spall [1997] and Spall and Cristion [1998], the one-simulation form has potential advantages in nonstationary systems wherein the underlying process dynamics change significantly during the course of a simulation. Our one-simulation algorithms also need to be tried in those type of settings. The deterministic perturbations that we proposed can also be applied to the one-timescale algorithms. Some experiments with certain test functions in one-timescale algorithms showed performance improvements over randomized perturbations.

5. CONCLUSIONS

Our goal was to investigate the use of deterministic perturbation sequences in order to enhance the convergence properties of SPSA algorithms, which have been found to be effective for attacking high-dimensional simulation optimization problems. Using two alternative constructions—lexicographic and Hadamard matrix-based—we developed many new variants of two-timescale SPSA algorithms, including one-simulation versions. For an N-dimensional parameter, the lexicographic construction requires a cycle of 2^{N-1} and 2^N points

for two- and one-simulation algorithms, respectively, whereas the Hadamard matrix-based construction requires only $2^{\lceil \log_2 N \rceil}$ and $2^{\lceil \log_2 (N+1) \rceil}$ points, respectively. Along with rigorous convergence proofs for all algorithms, numerical experiments on queueing networks with parameters of varying dimension indicate that the deterministic perturbation sequences do indeed show promise for significantly faster convergence.

Although we considered the setting of long-run average cost, the algorithms can easily be applied to the case of terminating (finite horizon, as opposed to steady-state) simulations where $\{X_j\}$ would be independent sequences (instead of a Markov process), each element representing the output of one independent replication, see Law and Kelton [2000]. Also, in our numerical experiments, we terminated all simulations after a fixed number of function evaluations, whereas in practical implementation, a stopping rule based on the sample data (e.g., estimated gradients) would be desirable. However, this problem is common to the application of all stochastic approximation algorithms, and was not a focus of our work.

There is obviously room for further exploration of the ideas presented here. Although the SPSA algorithms based on deterministic perturbation sequences appear to dominate their randomized difference counterparts empirically, there is no clear dominance between the two deterministic constructions. The impetus for the Hadamard matrix construction was to shorten the cycle length in order to reduce the aggregate bias, which we conjectured would lead to faster convergence properties. In the case of one-simulation algorithms, the Hadamard matrix construction did indeed show the best performance. However, in the case of two-simulation algorithms, this was not the case, as the Hadamard matrix-based algorithms showed a slight advantage in low-dimensional problems. In high-dimensional examples, the lexicographic-based algorithms performed better. Although we attempted to provide some plausible explanations for these outcomes, a more in-depth analysis would be useful. Also, in the implementation of all of the proposed algorithms, the same order within a cycle was used throughout the SA iterations, so randomizing the order of the points visited in a cycle might be worth investigating. Approaches from design of experiments methodology might provide ideas for implementations that are a hybrid of random and deterministic perturbation sequences. Finally, the connection with concepts and results from quasi-Monte Carlo warrants a deeper investigation.

ACKNOWLEDGMENTS

We thank the co-Editor Prof. Barry Nelson and two anonymous referees for their many comments and suggestions that have led to a significantly improved exposition.

REFERENCES

BHATNAGAR, S. 1997. Multiscale stochastic approximation algorithms with applications to ABR service in ATM networks. Ph.D. dissertation, Department of Electrical Engineering, Indian Institute of Science, Bangalore, India.

BHATNAGAR, S. AND BORKAR, V. S. 1997. Multiscale stochastic approximation for parametric optimization of hidden Markov models. *Prob. Eng. Inf. Sci. 11*, 509–522.

BHATNAGAR, S. AND BORKAR, V. S. 1998. A two timescale stochastic approximation scheme for simulation based parametric optimization. *Prob. Eng. and Inf. Sci. 12*, 519–531.

BHATNAGAR, S., FU, M. C., MARCUS, S. I., AND BHATNAGAR, S. 2001a. Two timescale algorithms for simulation optimization of hidden Markov models. *IIE Trans. 33*, 3, 245–258.

BHATNAGAR, S., FU, M. C., MARCUS, S. I., AND FARD, P. J. 2001b. Optimal structured feedback policies for ABR flow control using two-timescale SPSA. *IEEE/ACM Trans. Netw. 9*, 4, 479–491.

BORKAR, V. S. 1997. Stochastic approximation with two timescales. *Syst. Cont. Lett. 29*, 291–294.

CHEN, H. F., DUNCAN, T. E., AND PASIK-DUNCAN, B. 1999. A Kiefer-Wolfowitz algorithm with randomized differences. *IEEE Trans. Auto. Cont. 44*, 3, 442–453.

CHONG, E. K. P. AND RAMADGE, P. J. 1993. Optimization of queues using an infinitesimal perturbation analysis-based stochastic algorithm with general update times. *SIAM J. Cont. Optim. 31*, 3, 698–732.

CHONG, E. K. P. AND RAMADGE, P. J. 1994. Stochastic optimization of regenerative systems using infinitesimal perturbation analysis. *IEEE Trans. Auto. Cont. 39*, 7, 1400–1410.

FU, M. C. 1990. Convergence of a stochastic approximation algorithm for the $GI/G/1$ queue using infinitesimal perturbation analysis. *J. Optim. Theo. Appl. 65*, 149–160.

FU, M. C. AND HILL, S. D. 1997. Optimization of discrete event systems via simultaneous perturbation stochastic approximation. *IIE Trans. 29*, 3, 233–243.

GERENCSÉR, L. 1999. Rate of convergence of moments for a simultaneous perturbation stochastic approximation method for function minimization. *IEEE Trans. Auto. Cont. 44*, 894–906.

GERENCSÉR, L., HILL, S. D., AND VÁGÓ, Z. 1999. Optimization over discrete sets via SPSA. In *Proceedings of the IEEE Conference on Decision and Control* (Phoenix, Az.). IEEE Computer Society Press, Los Alamitos Calif., pp. 1791–1795.

HADAMARD, J. 1893. Résolution d'une question rélative aux determinants. *Bull. des Sciences Mathematiques 17*, 240–246.

HO, Y. C. AND CAO, X. R. 1991. *Perturbation Analysis of Discrete Event Dynamical Systems*. Kluwer, Boston.

KLEINMAN, N. L., SPALL, J. C., AND NAIMAN, D. Q. 1999. Simulation-based optimization with stochastic approximation using common random numbers. *Management Science 45*, 1570–1578.

KUSHNER, H. J. AND CLARK, D. S. 1978. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer Verlag, New York.

LAW, A. M. AND KELTON, W. D. 2000. *Simulation Modeling and Analysis*. McGraw-Hill, New York.

NIEDERREITER, N. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Philadelphia, Pa.

NIEDERREITER, N. 1995. New developments in uniform pseudorandom number and vector generation. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, (H. Niederreiter and P. J-S. Shiue, Eds.). Lecture Notes in Statistics, Springer, New York.

PARK, S. K. AND MILLER, K. W. 1988. Random number generators: good ones are hard to find. *Commun. ACM 31*, 10, 1192–1201.

SANDILYA, S. AND KULKARNI, S. R. 1997. Deterministic sufficient conditions for convergence of simultaneous perturbation stochastic approximation algorithms. In *Proceedings of the 9th INFORMS Applied Probability Conference* (Boston, Mass.).

SEBERRY, J. AND YAMADA, M. 1992. Hadamard matrices, sequences, and block designs. In *Contemporary Design Theory—A Collection of Surveys*, D. J. Stinson and J. Dintiz, Eds. Wiley, New York, pp. 431–560.

SPALL, J. C. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Auto. Cont. 37*, 3, 332–341.

SPALL, J. C. 1997. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica 33*, 109–112.

SPALL, J. C. AND CRISTION, J. A. 1998. Model-free control of nonlinear stochastic systems with discrete-time measurements. *IEEE Trans. Auto. Cont. 43*, 9, 1198–1210.

WANG, I.-J. AND CHONG, E. K. P. 1998. A deterministic analysis of stochastic approximation with randomized directions. *IEEE Trans. Auto. Cont. 43*, 12, 1745–1749.

Received November 2001; revised June 2002 and September 2002; accepted October 2002