Adaptive Multivariate Three-Timescale Stochastic Approximation Algorithms for Simulation Based Optimization

SHALABH BHATNAGAR
Indian Institute of Science, Bangalore

We develop in this article, four adaptive three-timescale stochastic approximation algorithms for simulation optimization that estimate both the gradient and Hessian of average cost at each update epoch. These algorithms use four, three, two, and one simulation(s), respectively, and update the values of the decision variable and Hessian matrix components simultaneously, with estimates based on the simultaneous perturbation methodology. Our algorithms use coupled stochastic recursions that proceed using three different timescales or step-size schedules. We present a detailed convergence analysis of the algorithms and show numerical experiments using all the developed algorithms on a two-node network of $M/G/1$ queues with feedback for a 50-dimensional parameter vector. We provide comparisons of the performance of these algorithms with two recently developed two-timescale *steepest descent* simultaneous perturbation analogs that use randomized and deterministic perturbation sequences, respectively. We also present experiments to explore the sensitivity of the algorithms to their associated parameters. The algorithms that use four and three simulations, respectively, perform significantly better than the rest of the algorithms.

Categories and Subject Descriptors: I.6.1 [**Simulation and Modeling**]: Simulation Theory; G.3.8 [**Probability and Statistics**]: Probabilistic Algorithms (including Monte Carlo); I.6.0 [**Simulation and Modeling**]: General

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Adaptive three-timescale stochastic approximation algorithms, simulation optimization, simultaneous perturbation stochastic approximation, Newton-type algorithms

1. INTRODUCTION

Simulation-based approaches for continuous valued parameter optimization have largely been studied using gradient-based algorithms. Among these, perturbation analysis (PA) type approaches (see for instance Chong and Ramadge

This work was supported in part by Grant number SR/S3/EE/43/2002-SERC-Engg. from the Department of Science and Technology, Government of India.
Author's address: S. Bhatnagar, Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India; email: shalabh@csa.iisc.ernet.in.

[1993]; Chong and Ramadge [1994]; Ho and Cao [1991]; Fu [1990]) assume knowledge of sample path gradients and typically use only one simulation. Likelihood ratio (LR)-based approaches (Andradóttir [1996], L'Ecuyer and Glynn [1994]) also use only one simulation, assuming knowledge of pathwise gradients, and rely on a change of measure with respect to which the average cost expectation is taken. Both PA and LR-based approaches, however, require constraining regularity conditions on the underlying system model and performance functions. Moreover, in many of these approaches, the parameter is updated only at certain regeneration epochs which can be sparse in practice.

Among stochastic approximation algorithms based on estimating gradients, the simultaneous perturbation stochastic approximation (SPSA) algorithm, introduced in Spall [1992], is applicable in a wide range of settings (see for instance, Bhatnagar et al. [2001a]; Chen et al. [1999]; Fu and Hill [1997]; Kleinman et al. [1999]) and is, in general, found to be particularly effective in cases where the parameter dimension is high. SPSA typically uses only two samples of the objective function and updates the entire parameter vector at each update epoch by randomly perturbing all parameter components simultaneously, unlike Kiefer-Wolfowitz (K-W) type algorithms (Kiefer and Wolfowitz [1952]) based on (symmetric) finite difference estimates that require $2N$ samples for an N-dimensional parameter vector. K-W algorithms with one-sided differences require $(N + 1)$ samples. Another version of SPSA, proposed in Spall [1997], uses only one sample but does not show as good performance as its two-simulation counterpart. Adaptive stochastic approximation algorithms, based on computing the Hessian (in addition to the gradient) estimates, typically require many more samples of the objective function than those that estimate only the gradient. For instance, in Fabian [1971], the Hessian is estimated using finite differences that are, in turn, based on finite difference estimates of the gradient. This requires $O(N^2)$ samples of the objective function at each update epoch. In Ruppert [1985], an adaptive multivariate version of the Robbins-Monro algorithm (Robbins and Monro [1951]) is obtained for the case where the objective function gradients are known and the Hessian is estimated using finite gradient differences. In Dippon and Renz [1997], algorithms that use gradient estimates based on certain weighted averages over a finite number of sample paths are shown to have similar asymptotic mean squared error as in Newton-type algorithms. Regular averages over all sample paths up to each update epoch are considered in Polyak and Juditsky [1992] and shown to improve performance.

Recently, in Spall [2000], a simultaneous perturbation-based Newton-type stochastic adaptive algorithm has been proposed. The Hessian estimates in this algorithm are obtained using four objective function samples at each update epoch in cases where the gradient estimates are not known, and three samples in cases where the latter are known. This is achieved by using two independent perturbation sequences with random variables in these that are assumed bounded, zero-mean, symmetric, and that have a common distribution and are mutually independent of one another. This method is an extension of the *steepest descent* SPSA algorithm of Spall [1992] that uses only one such perturbation sequence. The adaptive algorithms of Spall [2000] have been shown to perform better than SPSA under the settings considered in Spall [2000] and Luman [2000]. In order to ensure positive definiteness of the Hessian estimates, these are first projected at the parameter update step onto the set of positive definite matrices. Certain mappings for this purpose have been described in Spall [2000] (see also Bertsekas [1999]). In Zhu and Spall [2002], another such mapping based on projecting the eigenvalues of Hessian updates on the positive half line is given, and an algorithm that replaces the inverse of the projected Hessian update with that of the geometric mean of the projected eigenvalues in the parameter recursion step is found to show good performance.

In this article, we develop four adaptive algorithms for simulation-based parameter optimization for cases where both the gradient and Hessian are unknown and need to be estimated. These use four, three, two and one simulation(s), respectively, at each update epoch. All our algorithms work with three different step-size schedules, or timescales, and use coupled stochastic recursions. Even though these algorithms are based on the simultaneous perturbation concept, the form of the gradient estimate in the two-simulation (2SA) and one-simulation (1SA) algorithms is considerably different from corresponding two-simulation and one-simulation SPSA algorithms. We present a detailed convergence analysis of our four-simulation algorithm (4SA) where the above ideas are formalized, and describe the key changes required in the analysis for the rest of the algorithms. Algorithm 4SA uses a similar estimate for the Hessian as the corresponding algorithm of Spall [2000] except that we do not impose the constraint that each Hessian update be symmetric as Spall does. The same is true with the Hessian estimates in our other algorithms as well. We feel that the above symmetry constraint is not needed unless the projection mechanism itself requires that it be so, as, for instance, in the projection scheme described in Zhu and Spall [2002] where the symmetry of each Hessian update is required by the former. However, the Hessian estimates in our algorithms can easily be modified as in Spall [2000] to make them symmetric (if required) at each iteration and the convergence analysis that we present would still go through with minor changes. We show numerical comparisons of our algorithms on a two-node network of $M/G/1$ queues with feedback, with parameters of dimension 4 and 50, respectively. Note that projecting the Hessian estimates onto the set of positive definite matrices and obtaining their inverses typically involves a lot of computation. For this reason, in our experiments, we consider in place of the Hessian, a suitable diagonal matrix with each diagonal element in it representing the estimate of the second partial derivative of average cost with respect to the corresponding component of the decision variable. This is usually recommended for high-dimensional parameters (see for instance, Spall [2000]; Bertsekas [1999]) so as to keep in check the computational complexity of the procedure.

The rest of the article is organized as follows: In Section 2, we start with a brief overview of deterministic optimization algorithms and describe their stochastic analogs. We then present the framework and problem formulation, and also describe the SPSA based two-timescale algorithm of Bhatnagar et al.

[2001a]. We present our adaptive three-timescale algorithms in Section 3 and state the main convergence results. In Section 4, we first briefly describe the deterministic perturbation two-timescale SPSA algorithm of Bhatnagar et al. [2003]. Next, we present our numerical results using the various algorithms. Finally, in the Appendix, we present the detailed convergence analysis of Algorithm 4SA and the key changes required in the analysis for the other algorithms.

2. MODEL AND ALGORITHMS

2.1 Overview of Deterministic Optimization Algorithms

Most standard algorithms for deterministic optimization problems (see for instance, Bertsekas [1999]; Bertsekas and Tsitsiklis [1989]) in which the aim is to find a parameter $\theta^* \in \mathcal{R}^N$ that minimizes a continuously differentiable function $\hat{J} : \mathcal{R}^N \to \mathcal{R}$, require knowledge (either an exact computation or estimate) of the gradient $\nabla \hat{J}(\cdot)$. A typical algorithm is of the form

$$\hat{\theta}(n+1) = \hat{\theta}(n) - \gamma[D(\hat{\theta}(n))]^{-1}\nabla \hat{J}(\hat{\theta}(n)), \tag{1}$$

where $D(\hat{\theta}(n))$ is a positive definite $N \times N$ matrix, and $\gamma > 0$ is a given step-size parameter. Suppose for any vector $y \in \mathcal{R}^N$, y^T denotes its transpose. Given $\theta \in \mathcal{R}^N$ such that $\nabla \hat{J}(\theta) \neq 0$, any $x \in \mathcal{R}^N$ satisfying $x^T \nabla \hat{J}(\theta) < 0$ is a *descent* direction since the directional derivative $x^T \nabla \hat{J}(\theta)$ along the direction x is negative, and thus by a Taylor series expansion of $\hat{J}(\theta + \gamma x)$ around θ, one has

$$\hat{J}(\theta + \gamma x) = \hat{J}(\theta) + \gamma x^T \nabla \hat{J}(\theta) + o(\gamma),$$

which means that $\hat{J}(\theta + \gamma x) < \hat{J}(\theta)$ for γ sufficiently small. Now since $D(\hat{\theta}(n))$ is a positive definite matrix, both $D(\hat{\theta}(n))^T$ and $D(\hat{\theta}(n))^{-1}$ are positive definite matrices. Hence it is easy to see that $x = -D(\hat{\theta}(n))^{-1}\, \nabla \hat{J}(\hat{\theta}(n))$ is a descent direction. Algorithms that update along descent directions are also called *descent algorithms*. The following well known algorithms are special cases of (1):

(1) *Gradient Algorithm.* Here $D(\hat{\theta}(n)) = I$ (the N-dimensional identity matrix). This is also called the *steepest descent* algorithm since it updates strictly along the direction of negative gradient.
(2) *Jacobi Algorithm.* In this algorithm, $D(\hat{\theta}(n))$ is set to be an $N \times N$-diagonal matrix with its ith diagonal element $\nabla^2_{i,i} \hat{J}(\hat{\theta}(n))$. For $D(\hat{\theta}(n))$ to be a positive definite matrix in this case, it is easy to see that all elements $\nabla^2_{i,i} \hat{J}(\hat{\theta}(n))$, $i = 1, \ldots, N$, should be positive.
(3) *Newton Algorithm.* Here $D(\hat{\theta}(n))$ is chosen to be $\nabla^2 \hat{J}(\hat{\theta}(n))$, or the Hessian of $\hat{J}(\hat{\theta}(n))$.

The $D(\hat{\theta}(n))$ matrices in Jacobi and Newton algorithms, respectively, need not be positive definite, in general, and hence should be projected appropriately after each parameter update so as to ensure that the resulting matrices are positive definite (Bertsekas [1999] pp. 88–98). Note that both Jacobi and Newton algorithms require that the function $\hat{J}$ be twice continuously differentiable. With proper scaling provided by the $D(\hat{\theta}(n))$ matrix, the descent directions obtained using Jacobi and Newton algorithms are preferable to the one using gradient algorithm. However, computation of the inverse of the (projected) Hessian matrix at each iteration in the Newton algorithm can result in a significant computational overhead, in particular when the parameter dimension N is high. Computing the inverse in the case of the Jacobi algorithm is much simpler as $D(\cdot)$ is a diagonal matrix.

2.2 Stochastic Approximation Algorithms for Optimization

Consider now the problem of finding the zeroes of a function $F(\theta)$, with $\theta \in \mathcal{R}^N$, given certain 'noisy' observations/estimates $f(\theta, \xi)$ of $F(\theta)$, where ξ is a random variable such that the expectation $E[f(\theta, \xi)]$ (with respect to the distribution of ξ) equals $F(\theta)$. Consider now an infinite sequence of such observations such that the corresponding noise terms ξ_n, $n \geq 1$, are independent. The Robbins-Monro algorithm (2) (Robbins and Monro [1951]) converges to a parameter θ^* for which $F(\theta^*) = 0$.

$$\theta(n+1) = \theta(n) + a(n) f(\theta(n), \xi_n), \tag{2}$$

where $\{a(n)\}$ is a sequence of step-sizes that satisfy

$$\sum_{n=0}^{\infty} a(n) = \infty, \quad \sum_{n=0}^{\infty} a(n)^2 < \infty.$$

If, on the other hand, the aim is to find θ^* that minimizes $\hat{J}(\theta)$, then this problem can be seen to be equivalent to the one above by setting $F(\theta) = -\nabla \hat{J}(\theta)$. Direct measurements of $\nabla \hat{J}(\theta)$, in general, are not possible. However, in cases where perturbation analysis (PA) type schemes (Chong and Ramadge [1993, 1999]; Ho and Cao [1991]; Fu [1990]) apply, the sample pathwise gradients of the cost function are obtained, and under certain constraining requirements on the system parameters, an interchange between the gradient and expectation operators is shown. Likelihood ratio (LR)-based approaches (see for instance, Andradóttir [1996]; L'Ecuyer and Glynn [1994]) also rely on an interchange between the gradient and expectation operators and are applicable in this setting. The expected cost in these is typically written via a change of measure (the new measure being independent of the parameter) as the expectation of the product of cost and a likelihood ratio term. LR approaches also require regularity conditions on the system parameters and cost functions. In many of the above schemes, the algorithm is updated at certain regeneration epochs of the basic underlying process which can be sparse in practice. The Kiefer-Wolfowitz (K-W) algorithm (Kiefer and Wolfowitz [1952]) with forward difference estimates, on the other hand, estimates $\nabla \hat{J}(\theta)$ using two measurements (if θ is a scalar) $\bar{J}(\theta + \delta, \xi^1)$ and $\bar{J}(\theta, \xi^2)$, respectively, of the loss function $\hat{J}(\theta)$. Here ξ^1 and ξ^2 are independent of one another, and are such that

$$E_{\xi^i}[\bar{J}(\theta, \cdot)] = \hat{J}(\theta), \tag{3}$$

where the expectation is with respect to the distribution of ξ^i, $i = 1, 2$. Note that, in general, the distributions of ξ^1 and ξ^2 could be different as long as (3) is satisfied. The K-W algorithm with forward difference estimates is thus the same as (2), if the estimate of $\nabla\hat{J}(\theta(n))$ $(= -F(\theta(n)))$ has the form $(\bar{J}(\theta(n) + \delta, \xi^1(n)) - \bar{J}(\theta(n), \xi^2(n)))/\delta$, where $\{\xi^i(n)\}$, $i = 1, 2$, are two mutually independent sequences of independent random variables such that (3) holds (with $\xi^i(n)$ in place of ξ^i). As stated previously, one needs two loss function measurements for estimating the gradient of $\hat{J}(\theta)$, if θ is a scalar. For an N-dimensional θ, with $N > 1$, one needs $(N + 1)$ loss function measurements using the K-W algorithm with forward difference estimates as previously stated. K-W with symmetric differences requires $2N$ loss finction measurements. In Spall [1992], an alternative algorithm that uses only two loss function measurements at each parameter update epoch has been proposed. Here the estimate for the ith partial derivative of $\hat{J}(\theta)$, $i = 1, \ldots, N$, has the form $(\bar{J}(\theta(n) + \delta\triangle(n), \xi^+(n)) - \bar{J}(\theta(n) - \delta\triangle(n), \xi^-(n)))/2\delta\triangle_i(n)$, where $\bar{J}(\cdot, \xi^+(n))$ and $\bar{J}(\cdot, \xi^-(n))$ are noisy estimates of $\hat{J}(\cdot)$, with $\{\xi^+(n)\}$, and $\{\xi^-(n)\}$ being mutually independent sequences of independent random variables that satisfy (3) (with $\xi^w(n)$, $w = -, +$, in place of ξ^i). Also $\triangle_i(n)$ are most often taken to be mutually independent, mean-zero, ± 1-valued, Bernoulli distributed, random variables. Further, $\triangle(n)$ is the vector $\triangle(n) = (\triangle_1(n), \ldots, \triangle_N(n))^T$. More general conditions on $\triangle_i(n)$ are given in Section 2.3; see also Spall [1992] and Chen et al. [1999] for similar conditions. The above algorithms are of the *steepest descent* variety.

Among Newton type algorithms, as stated earlier, in Fabian [1971], an adaptive algorithm based on estimating both the gradient and Hessian is considered, wherein the latter is estimated using a set of differences of finite difference K-W type estimates. This, however, requires $O(N^2)$ samples at each iteration. In Spall [2000], using the simultaneous perturbation approach, the number of samples required to estimate both the gradient and Hessian at each iteration is just four, for any N. Here the estimate of the (k, l)'th component of the Hessian has the form $(4\delta_1\delta_2)^{-1}[(\triangle_k(n)\hat{\triangle}_l(n))^{-1} + (\triangle_l(n)\hat{\triangle}_k(n))^{-1}]$ $[\bar{J}(\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n), \xi^{++}(n)) - \bar{J}(\theta(n) + \delta_1\triangle(n), \xi^+(n)) - \bar{J}(\theta(n) - \delta_1\triangle(n) + \delta_2\hat{\triangle}(n), \xi^{-+}(n)) + \bar{J}(\theta(n) - \delta_1\triangle(n), \xi^-(n))]$, where $\triangle_k(n)$, $\hat{\triangle}_l(n)$, $k, l = 1, \ldots, N$, are mutually independent random variables as described previously. Further, $\triangle(n) = (\triangle_1(n), \ldots, \triangle_N(n))^T$ and $\hat{\triangle}(n) = (\hat{\triangle}_1(n), \ldots, \hat{\triangle}_N(n))^T$, respectively. Also $\{\xi^{++}(n)\}$, $\{\xi^+(n)\}$, $\{\xi^{-+}(n)\}$ and $\{\xi^-(n)\}$ are mutually independent sequences of independent random variables, and are such that (3) holds (with $\xi^w(n)$, $w = +, -, -+, ++$, in place of ξ^i). The estimate of the Hessian is then averaged across samples in the algorithm of Spall [2000] and projected onto the set of positive definite matrices, the inverse of which is then used in the parameter update step.

In Bhatnagar and Borkar [1997] and Bhatnagar and Borkar [1998], two-timescale versions of the K-W algorithm with one-sided differences were developed as alternatives to PA type schemes. Here the estimates $\bar{J}$ themselves correspond to the long-run average cost at given perturbed parameter updates. The advantage in these schemes is that one updates the parameter vector at certain deterministic epochs, as opposed to regenerative instants as with many

PA type schemes. This is achieved by using two timescales or step-size schedules in the algorithm. The disadvantage in the schemes of Bhatnagar and Borkar [1997] and Bhatnagar and Borkar [1998], however, lies in the number of simulations (that increases with the parameter dimension) needed for performing one parameter update. Thus, these schemes are not efficient in high-dimensional settings. In Bhatnagar et al. [2001a], the SPSA versions of the algorithms in Bhatnagar and Borkar [1997] and Bhatnagar and Borkar [1998] were developed. These resulted in significantly better performance compared to the latter algorithms. In Bhatnagar et al. [2001b], one of the SPSA variants of Bhatnagar et al. [2001a] was applied to a problem of finding closed loop feedback optimal policies in available bit rate (ABR) service in asynchronous transfer mode (ATM) networks. Recently, in Bhatnagar et al. [2003], the use of certain deterministic perturbation sequences in place of randomized one, is proposed for *steepest descent* two-timescale SPSA. It is observed that if one identifies an appropriate set of perturbations, and at each instant cyclically moves the perturbation sequence in a deterministic manner through this set, then this results in an improvement in performance of SPSA type algorithms. In this article, we develop adaptive three-timescale simultaneous perturbation based Newton-type algorithms that use randomized differences and estimate both the gradient and Hessian at each update step.

2.3 Framework and Problem Formulation

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 $C \stackrel{\triangle}{=} \prod_{i=1}^{N}[a_{i,\min}, a_{i,\max}]$. We also assume that for any given $\theta \in C$, the process $\{X_n\}$ is ergodic Markov. We constrain our algorithms to evolve within the set C by using certain projection operators. 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

$$J(\theta) = \lim_{l\to\infty} \frac{1}{l} \sum_{j=0}^{l-1} h(X_j). \tag{4}$$

Let $\triangle_i(n)$, $\hat{\triangle}_i(n)$, $n \geq 0, i = 1, \ldots, N$, be the perturbation random variables. Also let $\{a(n)\}$, $\{b(n)\}$, and $\{c(n)\}$ be three step-size sequences. We make the following assumptions.

Assumption (A). $J(\theta)$ is twice continuously differentiable and has bounded third derivative.

Assumption (B). The random variables $\triangle_i(n)$, $\hat{\triangle}_i(n)$, $n \geq 0$, $i = 1, \ldots, N$, are mutually independent, mean-zero, have a common distribution and satisfy $E[(\triangle_i(n))^{-2}]$, $E[(\hat{\triangle}_i(n))^{-2}] \leq \bar{K}$, for some $\bar{K} < \infty$.

Assumption (C). The step-size schedules $\{a(n)\}$, $\{b(n)\}$ and $\{c(n)\}$ satisfy

$$\sum_n a(n) = \sum_n b(n) = \sum_n c(n) = \infty, \quad \sum_n (a(n)^2 + b(n)^2 + c(n)^2) < \infty, \tag{5}$$

$$a(n) = o(c(n)) \text{ and } c(n) = o(b(n)), \tag{6}$$

respectively.

Assumption (A) is mainly a technical condition required for proving convergence using certain Taylor series expansions of average cost and is a standard requirement. As stated earlier, one requires random perturbations for simultaneously updating all parameter components. In most applications (as with our numerical experiments), one simply takes $\triangle_i(n)$, $\hat{\triangle}_i(n)$, $i \in \{1, \dots, N\}$, $n \geq 0$, to be independent, mean-zero, ± 1-valued, Bernoulli distributed random variables. Finally, from Assumption (C), note that the slowest timescale corresponds to $\{a(n)\}$, and the fastest to $\{b(n)\}$. This is because $a(n)$ goes to zero the fastest and $b(n)$ the slowest, among the three step-size schedules. This has an impact on the corresponding sizes of increments in these recursions. In particular, beyond some finite N_0 (i.e., $n \geq N_0$), the sizes of increments in the recursion corresponding to $\{a(n)\}$ would uniformly be the smallest, and those corresponding to $\{b(n)\}$ would (uniformly) be the largest among the three types of recursions even though the increments asymptotically diminish to zero in all of these. One expects, therefore, that the recursions corresponding to $\{b(n)\}$ would asymptotically track their (corresponding) stable equilibrium points, the fastest albeit with a possibly higher variance in their trajectories. The timescale corresponding to $\{c(n)\}$ is faster than the one corresponding to $\{a(n)\}$, but slower than that corresponding to $\{b(n)\}$. In the following algorithms, we further average over L epochs the iterates corresponding to the timescale $\{b(n)\}$, in effect leading to an even faster timescale over which averaging is done. The difference in timescales of the various recursions helps in obtaining appropriate algorithmic behavior in the following manner. Note that an update of the value of the decision variable (recursion corresponding to $\{a(n)\}$) requires the knowledge or estimate of both the gradient and the inverse of the projected Hessian of average cost. This justifies the need for data to be averaged faster than the computation of both the gradient and the Hessian of $J(\theta)$. Further, the Hessian update corresponding to a given value of the decision variable must itself have converged when viewed from the timescale on which the latter is updated.

Note that since the parameter takes values within the set $C = \prod_{i=1}^{N} [a_{i,\min}, a_{i,\max}]$, we project the parameter iterates in our algorithms to the set C after each update using projection operators defined as follows: for given $x \in \mathcal{R}$, let Π_i, $i = 1, \dots, N$, be the maps $\Pi_i : \mathcal{R} \to [a_{i,\min}, a_{i,\max}]$ defined by $\Pi_i(x) = \max(\min(a_{i,\max}, x), a_{i,\min})$. Then Π_i projects x to the interval $[a_{i,\min}, a_{i,\max}]$. Also for $y = (y_1, \dots, y_N)^T \in \mathcal{R}^N$, let $\Pi(y) = (\Pi_1(y_1), \dots, \Pi_N(y_N))^T$. Then Π projects $y \in \mathcal{R}^N$ to the set C. Let $\triangle(n) = (\triangle_1(n), \dots, \triangle_N(n))^T$ and $\hat{\triangle}(n) = (\hat{\triangle}_1(n), \dots, \hat{\triangle}_N(n))^T$, respectively. We now describe the two timescale *steepest descent*, randomized difference SPSA algorithm of Bhatnagar et al. [2001a] (referred to as SPSA-R here) that uses only two simulations at each instant.

2.4 Two-Timescale Randomized Difference SPSA (SPSA-R)

Let $\delta > 0$ be a given small constant. Suppose $\triangle_i(n)$, $i = 1, \dots, N$, $n \geq 0$ are random variables satisfying Assumption (B). Note that the perturbations $\hat{\triangle}(n)$, $n \geq 0$ are not required in this algorithm. Consider two parallel simulations $\{X^-(l)\}$

and $\{X^+(l)\}$, governed by 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 (see the recursions following). Thus, 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 $n \geq 0$, $m \in \{0, \ldots, L-1\}$,

$$Z^-(nL+m+1) = Z^-(nL+m) + b(n)(h(X^-(nL+m)) - Z^-(nL+m)), \quad (7)$$

$$Z^+(nL+m+1) = Z^+(nL+m) + b(n)(h(X^+(nL+m)) - Z^+(nL+m)). \quad (8)$$

Next for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Pi_i\left(\theta_i(n) + a(n)\left[\frac{Z^-(nL) - Z^+(nL)}{2\delta\triangle_i(n)}\right]\right). \quad (9)$$

Here $\{a(n)\}$ and $\{b(n)\}$ satisfy

$$\sum_n a(n) = \sum_n b(n) = \infty, \quad \sum_n (a(n)^2 + b(n)^2) < \infty, \quad a(n) = o(b(n)).$$

In the next section, we present our adaptive three-timescale algorithms.

3. ADAPTIVE THREE-TIMESCALE ALGORITHMS

The following algorithms that we present use four, three, two and one simulation(s), respectively, and estimate both the gradient, and the Hessian of average cost. To ensure that each update of the Hessian matrix is positive definite, we project the same, using an appropriate projection operator $\Gamma : \mathcal{R}^{N\times N} \rightarrow$ {positive definite matrices}. Note that in a 'small' neighborhood of a local minimum, the Hessian matrix is expected to be positive definite. However, it need not be so in other portions of the parameter space. We assume $\Gamma(A) = A$, if A is positive definite. In general, various operators described for instance via the *modified Choleski factorization procedure*, see Bertsekas [1999], or the ones presented in Spall [2000] and Zhu and Spall [2002], respectively, can be used for projecting the Hessian updates onto the space of positive definite matrices. We shall not go into the details of these procedures as they can be found in the mentioned references. Let $\{\Gamma(A)\}^{-1}$ denote the inverse of $\Gamma(A)$. We assume that the operator Γ satisfies the following:

Assumption (D). If $\{A_n\}$ and $\{B_n\}$ are sequences of matrices in $\mathcal{R}^{N\times N}$ such that $\lim_{n\to\infty} \parallel A_n - B_n \parallel = 0$, then $\lim_{n\to\infty} \parallel \Gamma(A_n) - \Gamma(B_n) \parallel = 0$ as well. Further, for any sequence $\{C_n\}$ of matrices in $\mathcal{R}^{N\times N}$, if $\sup_n \parallel C_n \parallel < \infty$, then $\sup_n \parallel \Gamma(C_n) \parallel$, $\sup_n \parallel \{\Gamma(C_n)\}^{-1} \parallel < \infty$ as well.

Here and in the rest of the article, for any vector $x \in \mathcal{R}^N$, $\parallel x \parallel$ denotes its Euclidean norm. Further for any matrix $A \in \mathcal{R}^{N\times N}$, its norm is defined as the induced matrix norm, also denoted using $\parallel \cdot \parallel$ and defined according to $\parallel A \parallel = \max_{\{x\in\mathcal{R}^N \mid \parallel x \parallel = 1\}} \parallel Ax \parallel$. Note that the continuity requirement on Γ can be easily imposed in the *modified Choleski factorization procedure* and the operators in Spall [2000]. Also the procedure in Zhu and Spall [2002] has been shown to satisfy this requirement. In fact, since $\| A_n - B_n \| \to 0$ as $n \to \infty$, the eigenvalues of A_n and B_n asymptotically become equal since they are themselves uniformly continuous functions of the elements of these matrices. A sufficient condition (Bertsekas [1999] pp.35) for the other requirements in Assumption (D) is that the eigenvalues of each projected Hessian update be bounded above and away from zero. Thus for some scalars $c_1, c_2 > 0$, suppose Γ is such that

$$c_1 \| z \|^2 \leq z^T \Gamma(C_n) z \leq c_2 \| z \|^2, \quad \forall z \in \mathcal{R}^N, \quad n \geq 0. \tag{10}$$

Then all eigenvalues of $\Gamma(C_n)$, $\forall n$, lie between c_1 and c_2. The above also ensures that the procedure does not get stuck at a nonstationary point. Now by Propositions A.9 and A.15 of Bertsekas [1999], $\sup_n \| \Gamma(C_n) \|$, $\sup_n \| \{\Gamma(C_n)\}^{-1} \| < \infty$. Most projection operators are seen to satisfy (10) either by explicitly projecting eigenvalues to the positive half line as with Zhu and Spall [2002], or by (10) getting automatically enforced as, for instance, in the *modified Choleski factorization* procedure, see Bertsekas [1999]. Moreover, with suitable modifications (see for instance of Bertsekas [1999] pp. 729–734), the mappings mentioned in Spall [2000] can also be seen to satisfy (10). A more general condition than (10) is, however, given on page 36 of Bertsekas [1999]. We show in Lemma A.6 that $\sup_n \| H(n) \| < \infty$ w.p. 1, where $H(n)$ is the nth update of the Hessian. Assumption (D) is a technical requirement and is needed in the convergence analysis. In Spall [2000] and Zhu and Spall [2002], the corresponding operators are denoted as f_n. We, however, drop the dependence on time index n of these operators and denote them simply using the symbol Γ as such dependence is often not required and may lead to confusion. Note that the matrices $f_n(C_n)$ in Zhu and Spall [2002] depend explicitly only on matrices C_n, and not n. Moreover, with suitable modifications (mentioned previously), the mappings in Spall [2000] can also be characterized using a unique map. Let $\delta_1, \delta_2 > 0$ be given constants. Also let $L \geq 1$ be a given integer. The convergence analysis of the algorithms that follow is given in detail in the Appendix. However, we state here the main convergence results.

3.1 Four-Simulation Algorithm (4SA)

Consider four parallel simulations $\{X^-(l)\}$, $\{X^+(l)\}$, $\{X^{-+}(l)\}$, and $\{X^{++}(l)\}$ that are governed by the parameter sequences $\{\theta(n) - \delta_1\triangle(n)\}$, $\{\theta(n) + \delta_1\triangle(n)\}$, $\{\theta(n) - \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)\}$, and $\{\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)\}$, respectively, where l and n are related according to $l = nL + m$, for some $m \in \{0, 1, \ldots, L-1\}$. Let $Z^w(nL+m)$, $w \in \{-, +, -+, ++\}$ be quantities defined by recursions (11)–(14) that are used for averaging the cost function in the four simulations. We initialize $Z^w(0) = 0$, $\forall w \in \{-, +, -+, ++\}$. The algorithm is given as follows:
For $n \geq 0$, $m = 0, 1, \ldots, L-1$,

$$Z^-(nL+m+1) = Z^-(nL+m) + b(n)(h(X^-(nL+m)) - Z^-(nL+m)), \tag{11}$$

$$Z^+(nL+m+1) = Z^+(nL+m) + b(n)(h(X^+(nL+m)) - Z^+(nL+m)), \tag{12}$$

$$Z^{-+}(nL+m+1) = Z^{-+}(nL+m) + b(n)(h(X^{-+}(nL+m)) - Z^{-+}(nL+m)), \tag{13}$$

$$Z^{++}(nL+m+1) = Z^{++}(nL+m) + b(n)(h(X^{++}(nL+m)) - Z^{++}(nL+m)). \tag{14}$$

For $j, i = 1, \ldots, N$,

$$H_{j,i}(n+1) = H_{j,i}(n) + c(n)\left(\frac{\left(\frac{Z^{++}(nL)-Z^{+}(nL)}{\delta_2\hat{\triangle}_j(n)}\right) - \left(\frac{Z^{-+}(nL)-Z^{-}(nL)}{\delta_2\hat{\triangle}_j(n)}\right)}{2\delta_1\triangle_i(n)} - H_{j,i}(n)\right). \tag{15}$$

Next, form the matrix $P(n) = \Gamma([[H_{k,l}(n)]]_{k,l=1}^N)$, and let $M(n) = [[M_{k,l}(n)]]_{k,l=1}^N$ be the inverse of $P(n)$. Finally, for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Pi_i\left(\theta_i(n) + a(n)\sum_{k=1}^{N} M_{i,k}(n)\left(\frac{Z^{-}(nL) - Z^{+}(nL)}{2\delta_1\triangle_k(n)}\right)\right). \tag{16}$$

An ordinary differential equation (ODE)-based approach is used for proving the convergence of this algorithm (as also the other algorithms that follow). We state here our main result with the detailed analysis given in the Appendix. For any bounded and continuous function $v(\cdot) : \mathcal{R} \to \mathcal{R}$, let

$$\tilde{\pi}_i(v(y)) = \lim_{0<\eta\to 0}\left(\frac{\pi_i(y+\eta v(y)) - \pi_i(y)}{\eta}\right).$$

Also for $x = (x_1, \ldots, x_N)^T$, let $\tilde{\pi}(x) = (\tilde{\pi}_1(x_1), \ldots, \tilde{\pi}_N(x_N))^T$. Suppose $\bar{M}(\theta) = \{\Gamma(\nabla^2 J(\theta))\}^{-1}$ denotes the inverse of the 'projected Hessian matrix' corresponding to parameter θ, and let $\bar{M}_{k,l}(\theta)$ be its (k,l)'th element. The operator $\tilde{\pi}(\cdot)$ forces (in some sense) the ODE to evolve within the constraint set C. Consider the following ODE:

$$\dot{\theta} = \tilde{\pi}(-\bar{M}(\theta(t))\nabla J(\theta(t))). \tag{17}$$

Let

$$K \overset{\triangle}{=} \{\theta \in C \mid \nabla J(\theta)^T \tilde{\pi}(-\bar{M}(\theta)\nabla J(\theta)) = 0\}.$$

Further, given $\eta > 0$, let $K^\eta = \{\theta \in C \mid\parallel \theta - \theta_0 \parallel\leq \eta, \theta_0 \in K\}$. Then K^η denotes the set of all points that are within a distance η from the set K. Suppose $\hat{K} = \{\theta \in C \mid \tilde{\pi}(-\bar{M}(\theta)\nabla J(\theta)) = -\bar{M}(\theta)\nabla J(\theta)\}$. It is easy to see that $C^o \subseteq \hat{K}$, where C^o is the interior of C. We have

THEOREM 3.1. *Given $\eta > 0$, there exists $\hat{\delta} > 0$, such that for all $\delta_1, \delta_2 \in (0, \hat{\delta}]$, the algorithm* (11)–(16) *converges to K^η with probability one.*

Remark 3.1. Note that for $\theta \in \hat{K} \cap K$, $\nabla J(\theta) = 0$ by positive definiteness of $\bar{M}(\theta)$. Further, on the set $K \backslash \hat{K}$, if $\nabla J(\theta) \neq 0$, one has $\tilde{\pi}_i(-(\bar{M}(\theta)\nabla J(\theta))_i) = 0$ for all those i $(i = 1, \ldots, N)$ for which $\nabla_i J(\theta) \neq 0$. (Here $-(\bar{M}(\theta)\nabla J(\theta))_i$ corresponds to the ith component of the vector $(\bar{M}(\theta)\nabla J(\theta))$.) The latter correspond to spurious fixed points that, however, can Occur only on the projection set boundaries (since $C^o \subseteq \hat{K}$) as with any projection based algorithm (Kushner and Yin [1997] pp. 79). Now note that $\bar{K} \equiv \{\theta \mid \nabla J(\theta) = 0\}$ constitutes the set of all Kuhn-Tucker points, and not just local minima. However, points in $\bar{K}$ that are not local minima shall correspond to unstable equilibria. In principle, the stochastic approximation scheme may get trapped in an unstable equilibrium. In Pemantle [1990], with noise assumed to be sufficiently 'omnidirectional' in addition, it is shown that convergence of stochastic approximation algorithms to unstable fixed points is not possible (see also Brandiere [1998] for conditions on avoidance of unstable equilibria that lie in certain *compact connected chain recurrent sets*). Avoidance of unstable equilibria can be ensured by using additional independent noise. However, in most practical scenarios, stochastic approximation algorithms are known to converge to a stable equilibrium even without additional noise. For our algorithm, by continuity of $J(\cdot)$, one then obtains an 'ϵ-local minimum'. This implies that the algorithm converges either to a local minimum, or to a point that is at a distance $\leq \epsilon$ from it. Next, note that Theorem 3.1 merely gives the existence of a $\hat{\delta} > 0$, for given $\epsilon > 0$, such that $\forall \delta_1, \delta_2 \in (0, \hat{\delta}]$, the algorithm converges to an ϵ-local minimum, but does not give the precise value of such a $\hat{\delta}$. In Section 4, we discuss the effects of the choices of different δ_1, δ_2, L, $\{a(n)\}$, $\{b(n)\}$, and $\{c(n)\}$, respectively, on the numerical performance of the algorithm on our setting. Finally, for obtaining a global minimum, one may use in addition, a 'slowly decreasing Gaussian noise' in the slow timescale recursion (16), as in simulated annealing algorithms, see, for instance, Gelfand and Mitter [1991].

Remark 3.2. Remark 3.1 holds for all our algorithms and not just 4SA.

3.2 Three-Simulation Algorithm (3SA)

Consider three parallel simulations $\{X^-(l)\}$, $\{X^+(l)\}$, and $\{X^{++}(l)\}$ that are governed by the parameter sequences $\{\theta(n) - \delta_1\triangle(n)\}$, $\{\theta(n) + \delta_1\triangle(n)\}$, and $\{\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)\}$, respectively, where l has the form $l = nL + m$ as before, with $m \in \{0, 1, \ldots, L-1\}$. Let $Z^w(nL+m)$, $w \in \{-, +, ++\}$ be quantities defined by recursions (18)–(20) for averaging the cost functions in the three simulations. Also, we initialize $Z^w(0) = 0$, $\forall w \in \{-, +, ++\}$. The algorithm is as follows:
For $n \geq 0$, $m = 0, 1, \ldots, L-1$,

$$Z^-(nL+m+1) = Z^-(nL+m) + b(n)(h(X^-(nL+m)) - Z^-(nL+m)), \tag{18}$$

$$Z^+(nL+m+1) = Z^+(nL+m) + b(n)(h(X^+(nL+m)) - Z^+(nL+m)), \tag{19}$$

$$Z^{++}(nL+m+1) = Z^{++}(nL+m) + b(n)(h(X^{++}(nL+m)) - Z^{++}(nL+m)). \tag{20}$$

For $j, i \in \{1, \ldots, N\}$,

$$H_{j,i}(n+1) = H_{j,i}(n) + c(n)\left(\frac{Z^{++}(nL) - Z^+(nL)}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} - H_{j,i}(n)\right). \tag{21}$$

Next, form the matrix $P(n) = \Gamma([[H_{k,l}(n)]]_{k,l=1}^N)$, and let $M(n) = [[M_{k,l}(n)]]_{k,l=1}^N$ be the inverse of $P(n)$. Finally, for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Pi_i\left(\theta_i(n) + a(n)\sum_{k=1}^N M_{i,k}(n)\left(\frac{Z^-(nL) - Z^+(nL)}{2\delta_1\triangle_k(n)}\right)\right). \tag{22}$$

Note the change in expression of the Hessian estimates that require only two simulations in this case. We have the following analog of Theorem 3.1 for Algorithm 3SA.

THEOREM 3.2. *Given $\eta > 0$, there exists $\hat{\delta} > 0$ such that for all $\delta_1, \delta_2 \in (0, \hat{\delta}]$, the algorithm* (18)–(22) *converges to the set K^η with probability one.*

3.3 Two-Simulation Algorithm (2SA)

Consider two parallel simulations $\{X^+(l)\}$ and $\{X^{++}(l)\}$ that are governed by the parameter sequences $\{\theta(n) + \delta_1 \triangle(n)\}$, and $\{\theta(n) + \delta_1 \triangle(n) + \delta_2 \hat{\triangle}(n)\}$, respectively, where l and n are related according to $l = nL + m$ as before, with $m \in \{0, 1, \ldots, L-1\}$. Let $Z^w(nL+m)$, $w \in \{+, ++\}$ be quantities defined by recursions (23)–(24) for averaging the cost functions in the two simulations. Also, we initialize $Z^w(0) = 0 \ \forall w \in \{+, ++\}$. The algorithm is as follows:
For $n \geq 0$, $m = 0, 1, \ldots, L-1$,

$$Z^+(nL+m+1) = Z^+(nL+m) + b(n)(h(X^+(nL+m)) - Z^+(nL+m)), \tag{23}$$

$$Z^{++}(nL+m+1) = Z^{++}(nL+m) + b(n)(h(X^{++}(nL+m)) - Z^{++}(nL+m)). \tag{24}$$

For $j, i \in \{1, \ldots, N\}$,

$$H_{j,i}(n+1) = H_{j,i}(n) + c(n)\left(\frac{Z^{++}(nL) - Z^+(nL)}{\delta_1 \delta_2 \triangle_i(n) \hat{\triangle}_j(n)} - H_{j,i}(n)\right). \tag{25}$$

Next, form the matrix $P(n) = \Gamma([[H_{k,l}(n)]]_{k,l=1}^N)$, and let $M(n) = [[M_{k,l}(n)]]_{k,l=1}^N$ be the inverse of $P(n)$. Finally, for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Pi_i\left(\theta_i(n) + a(n)\sum_{k=1}^{N} M_{i,k}(n)\left(\frac{Z^+(nL) - Z^{++}(nL)}{\delta_2 \hat{\triangle}_k(n)}\right)\right). \tag{26}$$

Note the difference in the gradient estimate from the usual simultaneous perturbation gradient estimate that arises since we do not generate the simulation $\{X^-(l)\}$ here. As with SPSA type gradient estimates, we show, however, in Theorem 3.3 (see Appendix for a proof) that the bias terms asymptotically vanish and one obtains the desired *descent* direction using this algorithm as well. An interesting observation is that the numerator terms in the gradient and Hessian update components are the same except for a negative sign in the gradient estimate (that is used for descent direction). We have

THEOREM 3.3. *Given $\eta > 0$, there exists $\hat{\delta} > 0$, such that for all $\delta_1, \delta_2 \in (0, \hat{\delta}]$, the algorithm* (23)–(26) *converges to the set K^η with probability one.*

3.4 One-Simulation Algorithm (1SA)

Here we use only one simulation for estimating both gradient and Hessian. Consider the simulation $\{X^{++}(l)\}$ that is governed by $\{\theta(n) + \delta_1 \triangle(n) + \delta_2 \hat{\triangle}(n)\}$, where l and n are related according to $l = nL + m$ as before, with $m \in \{0, 1, \ldots, L-1\}$. Let $Z^{++}(nL+m)$ be defined by recursion (27) for averaging the cost function corresponding to this simulation. We initialize $Z^{++}(0) = 0$. The algorithm is as follows:
For $n \geq 0$, $m = 0, 1, \ldots, L-1$,

$$Z^{++}(nL+m+1) = Z^{++}(nL+m) + b(n)(h(X^{++}(nL+m)) - Z^{++}(nL+m)). \tag{27}$$

For $j, i \in \{1, \ldots, N\}$,

$$H_{j,i}(n+1) = H_{j,i}(n) + c(n)\left(\frac{Z^{++}(nL)}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} - H_{j,i}(n)\right). \tag{28}$$

Next, form the matrix $P(n) = \Gamma([[H_{k,l}(n)]]_{k,l=1}^N)$, and let $M(n) = [[M_{k,l}(n)]]_{k,l=1}^N$ be the inverse of $P(n)$. Finally, for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Pi_i\left(\theta_i(n) - a(n)\sum_{k=1}^{N} M_{i,k}(n)\left(\frac{Z^{++}(nL)}{\delta_2\hat{\triangle}_k(n)}\right)\right). \tag{29}$$

Note the difference in the forms of the gradient and Hessian estimates from the previous algorithms. In the one-measurement form of SPSA considered in Spall [1997] and its variants (also considered) in Bhatnagar et al. [2003], the gradient has the form $\frac{Z^{+}(nL)}{\delta_1\triangle_k(n)}$ as opposed to $\frac{Z^{++}(nL)}{\delta_2\hat{\triangle}_k(n)}$ in our algorithm. The difference arises since we are estimating the Hessian as well using the same simulation in addition to gradient. As with Algorithm 2SA, note that the numerators of the gradient and Hessian estimate components are also the same except for a negative sign in the latter (to indicate descent). We have

THEOREM 3.4. *Given $\eta > 0$, there exists $\hat{\delta} > 0$, such that for all $\delta_1, \delta_2 \in (0, \hat{\delta}]$, the algorithm* (27)–(29) *converges to the set K^η with probability one.*

Remark 3.3. We also developed two other variants (that we do not present here) where we use similar gradient estimates as 3SA and 2SA, but the estimates of the Hessian in these are replaced by the ones in Algorithm 1SA. The numerical performance in these algorithms was found to be inferior in comparison to Algorithms 3SA and 2SA respectively.

Remark 3.4. In Bhatnagar and Borkar [2003], the use of some chaotic iterative sequences for random number generation has recently been proposed for generating perturbations in SPSA and certain smoothed functional algorithms that use Gaussian noise. The same could also be tried for the case of the higher-order algorithms proposed here.

4. NUMERICAL EXPERIMENTS

We begin with a brief description of a deterministic perturbation algorithm from Bhatnagar et al. [2003] (described as SPSA2-2L there) that we refer to as SPSA-D. This algorithm is similar to SPSA-R, but is based on lexicographically ordering the space of perturbations and cyclically moving the perturbation sequence through this space in a deterministic manner. In Bhatnagar et al. [2003], another construction based on Hadamard matrices is also proposed. Both SPSA-D, and its analog, based on Hadamard matrices are found to improve performance considerably over SPSA-R. Moreover, SPSA-D is found to perform better than the Hadamard matrix-based algorithm for high-dimensional settings (Bhatnagar et al. [2003]) and the two are almost similar in performance over lower dimensions. SPSA-D is also found to show better results than the other algorithms considered in Bhatnagar et al. [2003]. In our experiments, we show performance comparisons of our algorithms with both SPSA-D and SPSA-R.

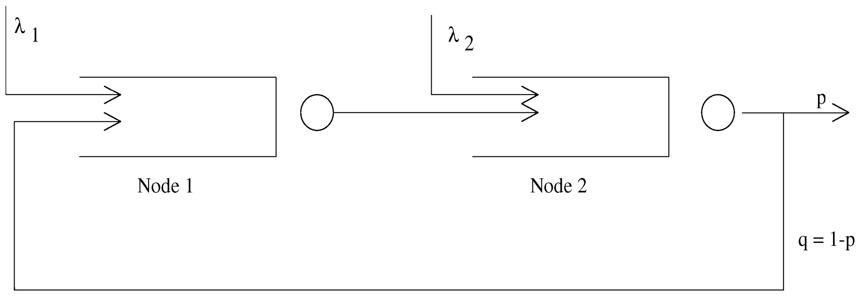

Fig. 1. Queueing network.

4.1 Two-Timescale Deterministic Perturbation Algorithm (SPSA-D)

We consider all perturbations $\triangle_i(n)$, $i = 1, \ldots, N$, $n \geq 0$, to be ± 1-valued. Suppose $E = \{\pm 1\}^N$ is the set of all perturbations. Note that the cardinality of E is 2^N. Fix $\triangle_1(n) = -1$, $\forall n \geq 0$ and lexicographically order all vectors in the resulting set $F \subset E$ of cardinality 2^{N-1}. Let $F = \{e_0, \ldots, e_{2^{N-1}-1}\}$. Thus each $e_i \in F$ is an N-dimensional vector with its first component set to -1. Now let $\triangle(0) = e_0$ and move the sequence cyclically through F. Algorithm SPSA-D is then given by recursions (7)–(9) of Algorithm SPSA-R, but with the sequence of perturbations as just noted.

4.2 Numerical Results

We consider a two-node network of $M/G/1$ queues with feedback as in Figure 1. The basic setting here is similar to that in Bhatnagar et al. [2001a] and Bhatnagar et al. [2003].

Nodes 1 and 2 are fed with independent Poisson arrival streams with rates $\lambda_1 = 0.2$, and $\lambda_2 = 0.1$, respectively. All customers after service at Node 1 enter Node 2. Further, after service completion at Node 2, each customer independently either leaves the system with probability $p = 0.4$, or joins Node 1 with probability $q = 0.6$. The service time processes $\{S_n^i(\theta^i)\}$ at the two nodes $i = 1, 2$ are defined by $S_n^i(\theta^i) = U_n^i(1 + \sum_{j=1}^M \sum_{j'=1}^M (\theta_j^i(n) - \bar{\theta}_j^i)(\theta_{j'}^i(n) - \bar{\theta}_{j'}^i)A_{j,j'})/R_i$, $i = 1, 2$, $n \geq 1$, where $U_n^1, U_n^2 \sim U(0, 1)$, $R_1 = 10$, and $R_2 = 20$, respectively. Further, $A = [[A_{j,j'}]]$ is a given positive definite and symmetric matrix. 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. 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 $\bar{\theta}_j^i = 0.3$, $j = 1, \ldots, M$, $i = 1, 2$. Also $\theta_j^1(0) = 0.4$, and $\theta_j^2(0) = 0.2$, $\forall j = 1, \ldots, M$. We assume the cost function to be the sum of waiting times of individual customers at the two nodes. Thus for the above cost to be minimized, the parameter components $\theta_k^i(n)$, $i = 1, 2$, $k = 1, \ldots, M$ should converge to $\bar{\theta}_k^i$ as $n \to \infty$. We show performance comparisons of the various algorithms on this setting.

We show our experiments for $M = 2$ and 25, that is, for parameters of dimension $N = 4$ and 50, respectively. We consider the Euclidean distance of each parameter update from the target parameter value $d(\theta(n), \bar{\theta}) \overset{\triangle}{=} (\sum_{i=1}^{2}\sum_{j=1}^{M}(\theta_j^i(n) - \bar{\theta}_j^i)^2)^{1/2}$ as our performance metric. For $M = 25$, we consider $A = I$ (the identity matrix), while for $M = 2$, we consider A with elements $A_{1,1} = A_{1,2} = A_{2,1} = 1$ and $A_{2,2} = 2$, respectively. Writing down the traffic equations for the above system (assuming stability for all θ-values), the net arrival rates at the two nodes (for both values of M) are $\gamma_1 = 0.65$ and $\gamma_2 = 0.75$, respectively. Further, the average service rate $\bar{\mu}_i$ at node i is bounded according to $1.739R_i \leq \bar{\mu}_i \leq 2R_i$, when $M = 2$ and $0.61R_i \leq \bar{\mu}_i \leq 2R_i$ when $M = 25$, respectively. The system is thus stable for both values of M and all values of θ. For the cost to be minimized, one expects $d(\theta(n), \bar{\theta}) \to 0$ as $n \to \infty$. For the randomized difference algorithms, we consider $\triangle_i(n)$, $\hat{\triangle}_i(n)$, $i = 1, \ldots, N$, $n \geq 0$ to be i.i.d., Bernoulli distributed taking values ± 1 with probability 1/2. In all our higher order algorithms considered here, we assume (for computational simplicity) the matrices $H(n)$ to have zero cross-diagonal terms, that is, $H_{k,l}(n) = 0$, $\forall k \neq l, n \geq 0$, with elements on the diagonal updated according to the recursions described in the various algorithms. This is usually recommended (Bertsekas [1999]; Spall [2000]) for high-dimensional settings. In order to ensure positive definiteness of the matrices $H(n)$, we simply project all diagonal terms $H_{k,k}(n)$, $k = 1, \ldots, N$, after each update to the interval $[0.1, \infty)$. Thus for implementation purposes, we use the Jacobi variants of the adaptive algorithms.

We now comment briefly on the validity of Assumptions (A)–(D) on this setting. In Bhatnagar [1997], certain sample path arguments using an application of dominated convergence theorem have been used to show that $J(\theta)$ is continuously differentiable with a bounded second derivative for a feedback controlled queuing system with general service times. The same arguments may further be extended and applied on the setting in this instance to show that the requirements in Assumption (A) hold. Assumption (B) holds since we use i.i.d., ± 1-valued Bernoulli random variables for the perturbations. Assumption (C) is clearly satisfied by the step-size sequences that we consider (see the following). Finally, it is easy to see that Assumption (D) is valid as well since projection is a continuous operator. Also (10) holds in this setting since all eigenvalues of the projected Hessian matrix take values greater than or equal to 0.1. Further, by Lemma A.6, the iterates of the Hessian (and hence of the projected Hessian) are uniformly upper bounded with probability one.

We terminate all simulation runs after 12×10^5 estimates of the cost function. The parameter is thus updated 3000 times for Algorithm 4SA, 4000 times for Algorithm 3SA, 6000 times for Algorithms 2SA, SPSA-R and SPSA-D, and 12000 times for Algorithm 1SA, at the end of these simulation runs. On a Pentium 5 PC with Linux operating system, each algorithm takes less than 30 seconds for one simulation run. We ran all simulations independently with twenty different initial seeds. In Figure 2, we plot the trajectories of the mean $d(\theta(n), \bar{\theta})$ obtained from the twenty independent simulation runs for $N = 50$ for Algorithms 4SA, 3SA, and SPSA-D with respect to the number of function evaluations. In Figure 3, we plot the same trajectories for Algorithms 2SA, 1SA, and SPSA-R. The convergence patterns for $N = 4$ are somewhat similar and are not shown to save space. The mean and standard error from all simulations upon termination for all algorithms are presented in Table I. We

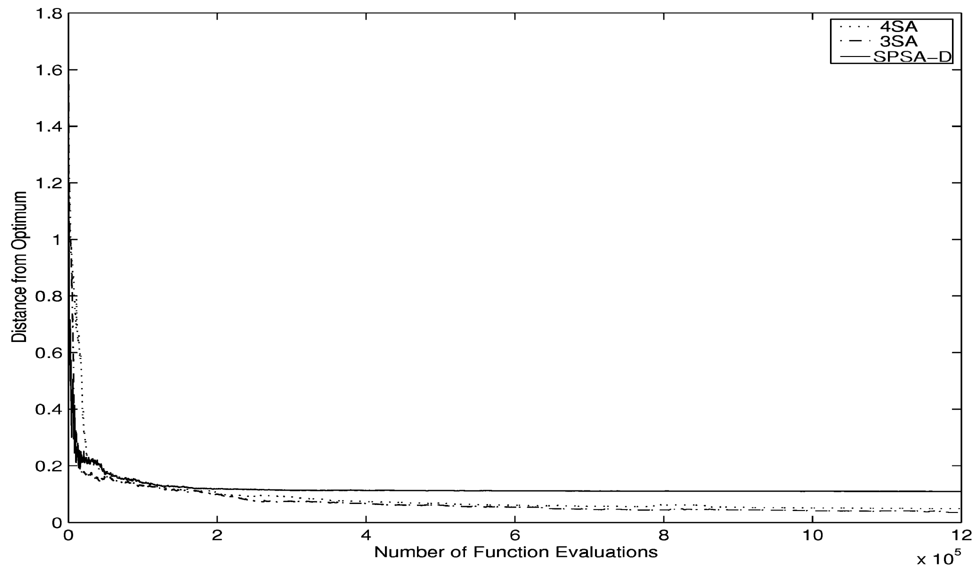

Fig. 2. Convergence behavior of Algorithms 4SA, 3SA and SPSA-D for $N = 50$.

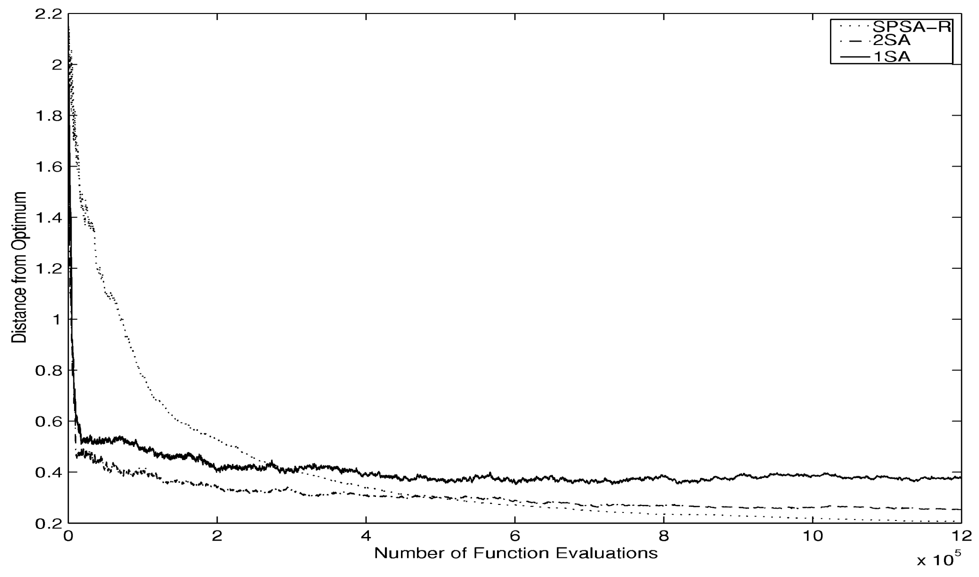

Fig. 3. Convergence behavior of Algorithms 2SA, 1SA and SPSA-R for $N = 50$.

select $L = 100$, $\delta_1 = \delta_2 = 0.1$ in all algorithms. Also $\{a(n)\}$, $\{b(n)\}$, and $\{c(n)\}$ are defined according to $a(n) = \frac{1}{n}$, $b(n) = \frac{1}{n^{2/3}}$, and $c(n) = \frac{1}{n^{3/4}}$, respectively, for $n \geq 1$, with $a(0) = b(0) = c(0) = 1$. For Algorithms SPSA-R and SPSA-D, $\{a(n)\}$, $\{b(n)\}$, and L are chosen as just shown. Further, δ is set at 0.1. From these experiments, we observe that Algorithms 4SA and 3SA show significantly better performance than the rest of the algorithms. For $N = 4$, 4SA performs better than 3SA, while for $N = 50$, 3SA is slightly better. Similar behavior is also

Table I. Performance After 12×10^5 Function Evaluations

Algorithm	$d(\theta(n), \bar{\theta})$ $N = 4$	$d(\theta(n), \bar{\theta})$ $N = 50$
4SA	0.0026 ± 0.0010	0.0488 ± 0.0182
3SA	0.0034 ± 0.0022	0.0364 ± 0.0149
2SA	0.0230 ± 0.0064	0.2537 ± 0.0330
1SA	0.1072 ± 0.0220	0.3768 ± 0.0228
SPSA-R	0.0343 ± 0.0199	0.2066 ± 0.0254
SPSA-D	0.0096 ± 0.0017	0.1093 ± 0.0134

Table II. Performance Variation of Algorithm 4SA With L

L	$d(\theta(n), \bar{\theta})$
1	0.4172 ± 0.0412
10	0.2140 ± 0.0214
50	0.1246 ± 0.0215
75	0.0471 ± 0.0138
100	0.0488 ± 0.0182
150	0.0328 ± 0.0099
200	0.0468 ± 0.0213
250	0.0408 ± 0.0247
300	0.0507 ± 0.0124
400	0.0632 ± 0.0322
500	0.0637 ± 0.0225
600	0.0732 ± 0.0381
700	0.0719 ± 0.0123
800	0.0876 ± 0.0463
900	0.1190 ± 0.05390
1000	0.1310 ± 0.0214
1200	0.1556 ± 0.0136

observed with varying δ_1 and δ_2 (see Table V and the discussion preceding it). It is likely that this behavior is observed because of the form of the A matrix in the two cases. Note that $A = I$ for $N = 50$, while A is a more general positive definite and symmetric matrix with nonzero cross diagonal terms for $N = 4$. Moreover, $H(n)$ is considered to be a diagonal matrix. $H(n)$, or A with nonzero cross diagonal terms, could result in 4SA showing better performance (over most cases) than 3SA, for $N = 50$ as well. SPSA-D shows better performance than SPSA-R, 2SA, and 1SA. It is interesting to observe that for $N = 4$, 2SA, is slightly better than SPSA-R. Algorithm 1SA does not show good performance. However, as observed in Spall [1997], the one-measurement form may have advantages in nonstationary settings. The same would also be true for 1SA.

Next, we study the effect of L, δ_1, δ_2 and the step-size sequences on performance. We show experiments where each of the twenty independent simulations are run for 12×10^5 estimates of the cost function value. In Table II, we study the variation of $d(\theta(n), \bar{\theta})$ for Algorithm 4SA, for $N = 50$, for different values of L, keeping the other parameters fixed as before. We observe that performance degrades for low and high L values. This is expected since for low L values, sufficient averaging is not achieved between two parameter updates,

Table III. Performance Variation of Algorithm 4SA With $c(n) = 1/n^{\alpha}$ for Fixed $a(n) = 1/n$ and $b(n) = 1/n^{0.55}$

α	$d(\theta(n), \bar{\theta})$
0.55	0.1572 ± 0.0359
0.60	0.1142 ± 0.04124
0.65	0.0572 ± 0.0147
0.70	0.0458 ± 0.0124
0.75	0.0427 ± 0.01564
0.80	0.0362 ± 0.0091
0.85	0.0364 ± 0.0101
0.90	0.0351 ± 0.0182
0.92	0.0331 ± 0.0204
0.96	0.1060 ± 0.0362
1.00	0.1322 ± 0.0298

Table IV. Performance Variation of Algorithm 4SA With $b(n) = 1/n^{\beta}$ for Fixed $a(n) = 1/n$ and $c(n) = 1/n^{0.75}$

β	$d(\theta(n), \bar{\theta})$
0.55	0.0417 ± 0.0164
0.58	0.0511 ± 0.0113
0.64	0.0291 ± 0.0193
0.66	0.0488 ± 0.0182
0.70	0.0335 ± 0.0144
0.72	0.0786 ± 0.0235
0.75	0.1126 ± 0.0368

while as L is increased, the number of parameter updates get reduced correspondingly, implying that excessive additional averaging is also not desirable. It has been observed in Bhatnagar et al. [2001a] that SPSA algorithms do not show good performance for very low L values (say $L \leq 10$) particularly for high-dimensional parameters. It appears from Table II that it is desirable to operate the algorithm ideally between $L = 75$ and $L = 300$. Similar behavior as in Table II is expected of the other algorithms as well.

In Table III, we set $\delta_1 = \delta_2 = 0.1$, $L = 100$, $a(n) = 1/n$, $b(n) = 1/n^{0.55}$ and $c(n) = 1/n^{\alpha}$, and study the effect of different α on the performance of 4SA for $N = 50$. We observe that the performance deteriorates when α is close to 0.55 (i.e., when the Hessian is updated on a similar timescale as data) or 1.00 (when the Hessian and the value of the decision variable are updated on a similar scale), while it is good in the range $\alpha \in [0.65, 0.92]$. This is again along expected lines. One expects a similar behavior for the other algorithms as well. Next, in Table IV, we set $\delta_1 = \delta_2 = 0.1$, $L = 100$, $a(n) = 1/n$, $c(n) = 1/n^{0.75}$ and $b(n) = 1/n^{\beta}$, and study the effect of different β on the performance of 4SA for $N = 50$. We observe that performance degrades when β is brought close to 0.75 (same scale on which the Hessian is updated). This also suggests the need for a clear separation between the three timescales. One expects a similar performance behavior with the other algorithms for this case as well.

Table V. Performance Variation of Algorithms 4SA and 3SA With δ_1 and δ_2

δ_1	δ_2	$d(\theta(n),\bar{\theta})$ for 4SA, $N=4$	$d(\theta(n),\bar{\theta})$ for 3SA, $N=4$	$d(\theta(n),\bar{\theta})$ for 4SA, $N=50$	$d(\theta(n),\bar{\theta})$ for 3SA, $N=50$
0.01	0.01	0.0166 ± 0.0042	0.0550 ± 0.0098	0.2688 ± 0.0427	0.2972 ± 0.0299
0.05	0.01	0.0102 ± 0.0051	0.0194 ± 0.0059	0.1927 ± 0.0213	0.1655 ± 0.0212
0.10	0.01	0.0062 ± 0.0013	0.0054 ± 0.0022	0.1447 ± 0.0199	0.1293 ± 0.0168
0.15	0.01	0.0060 ± 0.0023	0.0076 ± 0.0032	0.1444 ± 0.0146	0.1380 ± 0.0289
0.01	0.05	0.0028 ± 0.0010	0.0173 ± 0.0061	0.2012 ± 0.0401	0.2334 ± 0.0513
0.01	0.10	0.0112 ± 0.0083	0.0236 ± 0.0108	0.1990 ± 0.0382	0.2129 ± 0.0277
0.01	0.15	0.0369 ± 0.0128	0.0165 ± 0.0087	0.1375 ± 0.0185	0.1687 ± 0.0211
0.05	0.05	0.0055 ± 0.0013	0.0079 ± 0.0092	0.1262 ± 0.0220	0.1410 ± 0.0315
0.05	0.10	0.0030 ± 0.0014	0.0034 ± 0.0017	0.0561 ± 0.0074	0.0368 ± 0.0101
0.05	0.15	0.0032 ± 0.0022	0.0038 ± 0.0028	0.0464 ± 0.0102	0.0442 ± 0.0098
0.10	0.05	0.0045 ± 0.0011	0.0057 ± 0.0028	0.0416 ± 0.0119	0.0177 ± 0.0063
0.15	0.05	0.0047 ± 0.0022	0.0043 ± 0.0022	0.0469 ± 0.0144	0.0380 ± 0.0158
0.10	0.10	0.0026 ± 0.0010	0.0034 ± 0.0022	0.0488 ± 0.0182	0.0364 ± 0.0149
0.10	0.15	0.0040 ± 0.0028	0.0048 ± 0.0011	0.0274 ± 0.0112	0.0353 ± 0.0132
0.15	0.10	0.0040 ± 0.0020	0.0032 ± 0.0011	0.0569 ± 0.0214	0.0273 ± 0.0092

Finally, in Table V, we study the performance of both 4SA and 3SA for both $N = 4$ and $N = 50$, for different values of δ_1 and δ_2, with $L = 100$ and the step-sizes, the same as those used in Table I. We observe that for $N = 4$ ($N = 50$), 4SA shows better results than 3SA in eleven (six) out of the fifteen cases shown, with 3SA showing better results in the rest. Algorithm 4SA shows the best performance for $\delta_1 = 0.10$ and $\delta_2 = 0.10$ for $N = 4$ ($\delta_1 = 0.10$ and $\delta_2 = 0.15$ for $N = 50$) while 3SA gives the best results for $\delta_1 = 0.15$ and $\delta_2 = 0.10$ for $N = 4$ ($\delta_1 = 0.10$ and $\delta_2 = 0.05$ for $N = 50$). Note that the form of the gradient estimate is the same in both Algorithms 4SA and 3SA. The difference, however, lies in the estimates of the Hessian. It can be seen from the analysis in the Appendix that the bias in Hessian estimates of Algorithm 4SA is contributed to by the terms

$$\sum_{l\neq i} \frac{\triangle_l(n)}{\triangle_i(n)} \nabla^2_{j,l} J(\theta(n)), \quad \sum_{k\neq j} \frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)} \nabla^2_{k,i} J(\theta(n)), \tag{30}$$

$$\sum_{k\neq j}\sum_{l\neq i} \frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)}\frac{\triangle_l(n)}{\triangle_i(n)}\nabla^2_{k,l}J(\theta(n)), \quad \delta_2\sum_{k,l,m=1}^{N} \frac{\hat{\triangle}_k(n)\triangle_m(n)\nabla^3_{k,l,m}J(\theta(n))\hat{\triangle}_l(n)}{2\hat{\triangle}_j(n)\triangle_i(n)}, \tag{31}$$

while the same in Algorithm 3SA is contributed to by the terms

$$\sum_{l=1}^{N} \frac{1}{\delta_1}\frac{\hat{\triangle}_l(n)}{\triangle_i(n)\hat{\triangle}_j(n)}\nabla_l J(\theta(n)), \quad \sum_{l=1,l\neq j}^{N}\sum_{k=1,k\neq i}^{N} \frac{\hat{\triangle}_l(n)\triangle_k(n)}{\hat{\triangle}_j(n)\triangle_i(n)}\nabla^2_{l,k}J(\theta(n)), \tag{32}$$

$$\frac{\delta_1}{2}\sum_{l,k,m=1}^{N} \frac{\hat{\triangle}_l(n)\triangle_k(n)\triangle_m(n)}{\triangle_i(n)\hat{\triangle}_j(n)}\nabla^3_{k,m,l}J(\theta(n)), \quad \frac{\delta_2}{2\delta_1}\sum_{l,m=1}^{N}\frac{\hat{\triangle}_l(n)\hat{\triangle}_m(n)}{\triangle_i(n)\hat{\triangle}_j(n)}\nabla^2_{l,m}J(\theta(n)), \tag{33}$$

$$\frac{\delta_2}{2}\sum_{l=1}^{N}\sum_{m=1}^{N}\sum_{k=1}^{N}\frac{\hat{\triangle}_l(n)\hat{\triangle}_m(n)\triangle_k(n)}{\triangle_i(n)\hat{\triangle}_j(n)}\nabla^3_{l,m,k}J(\theta(n)). \tag{34}$$

Note that under Assumption (B), the conditional expectation of each of these terms, given $\theta(n)$, equals zero. An algorithm would be computationally superior if its updates have less contribution from terms of the this type, since the aforementioned terms contribute to the error in the Hessian estimates. Note that the contribution of the term in (34) is similar (by Assumption (B)) to that of the second term in (31). Also the contribution of the second term in (32) is similar to that of the first term in (31). Thus the main difference in bias lies in the two terms in (30) for the case of Algorithm 4SA, as opposited to the first term in (32) and the terms in (33) for the case of Algorithm 3SA. The first term in (32), however, can potentially contribute to a large bias in the Hessian estimates if δ_1 is small. This is also observed from Table I (see entries corresponding to $\delta_1 = 0.01$ for $N = 4$, and $\delta_1 = 0.01, 0.05$ for $N = 50$, for 3SA). As already stated, the slight edge in performance of 3SA over 4SA for $N = 50$ is possibly because of the form of the matrices $H(n)$ and A considered here. If one were to consider more general matrices, rather than diagonal ones, 4SA might perform better than 3SA for $N = 50$ as well. We also performed experiments with different δ_1, δ_2, and δ for the other algorithms and observed that 4SA and 3SA showed better results than the rest of the algorithms in most cases. Algorithms 2SA and 1SA exhibit greater bias in their gradient estimates as compared to Algorithms 4SA, 3SA, and SPSA-R. In particular, 2SA has less bias terms than 1SA.

The deterministic algorithms of Bhatnagar et al. [2003] require only one perturbation sequence $\{\triangle(n)\}$. The bias terms in the gradient estimates in these algorithms are thus relatively simple and, in particular, for two-simulation algorithms of the type SPSA-D, one requires for convergence that $\sum_{n=1}^{P}\frac{\triangle_k(n)}{\triangle_i(n)} = 0$, for all $k \neq i$, where P is the number of elements in the space of perturbations. However, construction of appropriate deterministic perturbation sequences for higher-order simultaneous perturbation algorithms is complicated because of the presence of two perturbation sequences $\{\triangle(n)\}$ and $\{\hat{\triangle}(n)\}$, respectively, and the need for similar conditions on bias terms in the Hessian estimates as in (30)–(31) for Algorithm 4SA ((32)–(34) for Algorithm 3SA), in addition to those in the gradient. The requirement that each of these should vanish asymptotically makes the task difficult. Algorithms 2SA and 1SA have even more numbers of terms in their corresponding expressions for bias in both Hessian and gradient. Clearly as one moves towards algorithms with a lower number of simulations, the number of terms contributing towards bias increase and more stringent conditions are needed for deterministic perturbation sequences to work in the case of higher-order algorithms. Construction of appropriate deterministic perturbation sequences for higher-order algorithms is an open problem.

APPENDIX

We present here the detailed convergence analysis of Algorithm 4SA and the key changes required in the analysis of the other algorithms.

A.1 Convergence Analysis of 4SA

Let $\mathcal{G}(l) = \sigma(\tilde{\theta}_i(p), \tilde{\triangle}_i(p), \tilde{\hat{\triangle}}_i(p), H_{j,i}(p), X^-(p), X^+(p), X^{-+}(p), X^{++}(p), p \leq l, i, j = 1, \ldots, N)$, $l \geq 1$, denote σ-fields generated by the quantities described. Here $\tilde{\theta}_i(p) = \theta_i(n)$, $\tilde{\triangle}_i(p) = \triangle_i(n)$, and $\tilde{\hat{\triangle}}_i(p) = \hat{\triangle}_i(n)$, respectively, for $i = 1, \ldots, N$, $nL \leq p \leq (n+1)L - 1$. Define $\{\tilde{b}(n)\}$ as follows: For $n \geq 0$, $\tilde{b}(n) = b([\frac{n}{L}])$, where $[\frac{n}{L}]$ denotes the integer part of $\frac{n}{L}$. Note that $\{\tilde{b}(n)\}$ corresponds to the natural timescale over which the data averaging step should be analyzed. It is easy to see that

$$\sum_n \tilde{b}(n) = \infty, \;\; \sum_n \tilde{b}(n)^2 < \infty, \;\; c(n) = o(\tilde{b}(n)). \tag{35}$$

In fact, $\{b(n)\}$ goes to zero faster than $\{\tilde{b}(n)\}$ does, and thus $\{\tilde{b}(n)\}$ corresponds to an even faster step-size sequence than $\{b(n)\}$. As a result, the additional averaging (over L epochs) of cost for the different simulations is seen to improve performance. Note that recursions (11)-(14) can be rewritten as

$$Z^w(p+1) = Z^w(p) + \tilde{b}(p)(h(X^w(p)) - Z^w(p)), \tag{36}$$

$w \in \{-, +, -+, ++\}$, with simulations $X^-(p)$, $X^+(p)$, $X^{-+}(p)$ and $X^{++}(p)$ governed by $\tilde{\theta}(p) - \delta_1\tilde{\triangle}(p)$, $\tilde{\theta}(p) + \delta_1\tilde{\triangle}(p)$, $\tilde{\theta}(p) - \delta_1\tilde{\triangle}(p) + \delta_2\tilde{\hat{\triangle}}(p)$, and $\tilde{\theta}(p) + \delta_1\tilde{\triangle}(p) + \delta_2\tilde{\hat{\triangle}}(p)$, respectively. Now define sequences $\{M^w(p)\}$, $w \in \{-, +, -+, ++\}$, as follows:

$$M^w(p) = \sum_{m=1}^{p} \tilde{b}(m)(h(X^w(m)) - E[h(X^w(m)) \mid \mathcal{G}(m-1)]).$$

It is easy to see that $\{M^w(p), \mathcal{G}(p)\}$ are martingale sequences. Also using (35), one can easily check that these individually converge almost surely.

Define $\{s(n), n \geq 0\}$ as follows: $s(0) = 0$, $s(n) = \sum_{i=0}^{n-1} a(i)$, $n \geq 1$. For $i = 1, \ldots, N$, let $\triangle_i(t) = \triangle_i(n)$ and $\hat{\triangle}_i(t) = \hat{\triangle}_i(n)$ for $t \in [s(n), s(n+1)]$, $n \geq 0$. Further let $\triangle(t) = (\triangle_1(t), \ldots, \triangle_N(t))^T$ and $\hat{\triangle}(t) = (\hat{\triangle}_1(t), \ldots, \hat{\triangle}_N(t))^T$, respectively. Now define $\{t(n)\}$ as follows: $t(0) = 0$, $t(n) = \sum_{i=0}^{n-1} \tilde{b}(i)$, $n \geq 1$. Consider the following system of ordinary differential equations (ODEs): For $i, j \in \{1, \ldots, N\}$, $w \in \{-, +, -+, ++\}$,

$$\dot{\theta}_i(t) = 0, \tag{37}$$

$$\dot{H}_{j,i}(t) = 0, \tag{38}$$

$$\dot{Z}^w(t) = J(\theta^w(t)) - Z^w(t). \tag{39}$$

In (39) and the rest of the article, we denote $\theta^-(t) = (\theta(t) - \delta_1\triangle(t))$, $\theta^+(t) = (\theta(t) + \delta_1\triangle(t))$, $\theta^{-+}(t) = (\theta(t) - \delta_1\triangle(t) + \delta_2\hat{\triangle}(t))$ and $\theta^{++}(t) = (\theta(t) + \delta_1\triangle(t) + \delta_2\hat{\triangle}(t))$, respectively. Before we proceed further, we recall a key result from Hirsch [1989] stated as Lemma A.1 as follows. Consider an ODE

$$\dot{x}(t) = F(x(t)), \tag{40}$$

which has an asymptotically stable attracting set G. Let G^ϵ denote the $\epsilon-$neighborhood of G that is, $G^\epsilon = \{x \mid \exists x' \in G \text{ s.t. } \| x - x' \| \leq \epsilon\}$. For $\tau > 0$,

$\mu > 0$, we call $y(\cdot)$ a (τ, μ)-perturbation of (40), if there exists an increasing sequence $\{\tau_i, i \geq 0\}$ of real numbers with $\tau_0 = 0$ and $\forall i$, $\tau_{i+1} - \tau_i \geq \tau$, such that on each interval $[\tau_i, \tau_{i+1}]$, there exists a solution $x^i(\cdot)$ of (40) such that $\sup_{t\in[\tau_i,\tau_{i+1}]} |x^i(t) - y(t)| < \mu$. We have

LEMMA A.1. *Given $\epsilon > 0$, $\tau > 0$, there exists a $\bar{\mu} > 0$ such that for all $\mu \in [0, \bar{\mu}]$, any (τ, μ)-perturbation of* (40) *converges to G^ϵ.*

Next, we have:

LEMMA A.2. *The iterates $Z^w(p)$, $\forall w \in \{-, +, -+, ++\}$, defined as in* (36) *are uniformly bounded with probability one.*

PROOF. Observe that $Z^w(p)$ are convex combinations from some finite n onwards of $Z^w(p-1)$ and a bounded quantity (since the cost function is bounded). The claim follows. □

Consider now functions $\bar{z}^w(t)$, $w \in \{-, +, -+, ++\}$, defined according to $\bar{z}^w(t(n)) = Z^w(nL)$ with the maps $t \to \bar{z}^w(t)$ corresponding to continuous linear interpolations on intervals $[t(n), t(n+1)]$. Given $T > 0$, define $\{T_n\}$ as follows: $T_0 = 0$ and for $n \geq 1$, $T_n = \min\{t(m) \mid t(m) \geq T_{n-1} + T\}$. Let $I_n = [T_n, T_{n+1}]$. Note that there exists some integer $m_n > 0$ such that $T_n = t(m_n)$. Define also functions $z^{w,n}(t)$, $w \in \{-, +, -+, ++\}$, $t \in I_n$, $n \geq 0$, according to

$$\dot{z}^{w,n}(t) = J(\theta^w(t)) - z^{w,n}(t), \tag{41}$$

with $z^{w,n}(T_n) = \bar{z}^w(t(m_n)) = Z^w((m_n)L)$. Now a routine argument, based on Gronwall's inequality and the fact that $\{M^w(n)\}$ is almost surely convergent, can be used to show (see Bhatnagar et al. [2001a])

LEMMA A.3. $\lim_{n\to\infty} \sup_{t\in I_n} \| z^{w,n}(t) - \bar{z}^w(t) \| = 0$ $\forall w \in \{-, +, -+, ++\}$, *w.p.*1.

Next, we have

LEMMA A.4. *Given $T, \gamma > 0$, $((\theta_i(t(n) + \cdot), H_{j,i}(t(n) + \cdot), \bar{z}^w(t(n) + \cdot))$, $i, j \in \{1, \ldots, N\}$, $w \in \{-, +, -+, ++\}$, is a bounded (T, γ)-perturbation of* (37)–(39) *for n sufficiently large.*

PROOF. Observe that the iterations (15)–(16) of the algorithm can be written as

$$H_{j,i}(n+1) = H_{j,i}(n) + \tilde{b}(n)\xi_1(n),$$

$$\theta_i(n+1) = \Pi_i(\theta_i(n) + \tilde{b}(n)\xi_2(n)),$$

respectively, where $\xi_1(n)$ and $\xi_2(n)$ are both $o(1)$ since $c(n)$, and $a(n)$ are individually $o(\tilde{b}(n))$. The rest now follows from Lemma A.3. □

COROLLARY A.5. *For all $w \in \{-, +, -+, ++\}$,*

$$\| Z^w(nL) - J(\theta^w(n)) \| \to 0 \ \ a.s.,$$

as $n \to \infty$.

PROOF. The claim follows by Lemma A.1 applied on (39) for every $\epsilon > 0$. □

We now concentrate on recursion (15) that updates the Hessian matrix components. We have the following important result:

LEMMA A.6. *The iterates $H_{j,i}(n)$, $n \geq 0$, $j, i \in \{1, \ldots, N\}$, in (15), are uniformly bounded with probability one.*

PROOF. Observe as in Lemma A.2 that from some finite n onwards, $H_{j,i}(n)$ is a convex combination of $H_{j,i}(n-1)$ and a uniformly bounded quantity. The claim follows. □

Let $F_{j,i}(\theta(n), \triangle(n), \hat{\triangle}(n))$ denote $\frac{(\frac{J(\theta^{++}(n))-J(\theta^{+}(n))}{\delta_2\hat{\triangle}_j(n)})-(\frac{J(\theta^{-+}(n))-J(\theta^{-}(n))}{\delta_2\hat{\triangle}_j(n)})}{2\delta_1\triangle_i(n)}$. Also let $\mathcal{F}(n) = \sigma(\theta_i(m), H_{j,i}(m), Z^{-}(mL), Z^{+}(mL), Z^{-+}(mL), Z^{++}(mL), m \leq n, i, j = 1, \ldots, N;$ $\triangle(m), \hat{\triangle}(m), m < n), n \geq 1$. Define sequences $\{N_{j,i}(n)\}, j, i = 1, \ldots, N$, according to

$$N_{j,i}(n) = \sum_{m=0}^{n-1} c(m)(F_{j,i}(\theta(m), \triangle(m), \hat{\triangle}(m)) - E[F_{j,i}(\theta(m), \triangle(m), \hat{\triangle}(m)) \mid \mathcal{F}(m)]).$$

It can be easily verified that $\{N_{j,i}(n), \mathcal{F}(n)\}$, $j, i = 1, \ldots, N$ form martingale sequences that are almost surely convergent. We now have

PROPOSITION A.7. *With probability one, $\forall j, i \in \{1, \ldots, N\}$,*

$$\left| E\left[\frac{\left(\frac{J(\theta^{++}(n))-J(\theta^{+}(n))}{\delta_2\hat{\triangle}_j(n)}\right) - \left(\frac{J(\theta^{-+}(n))-J(\theta^{-}(n))}{\delta_2\hat{\triangle}_j(n)}\right)}{2\delta_1\triangle_i(n)} \mid \mathcal{F}(n) \right] - \nabla^2_{j,i} J(\theta(n)) \right|$$

$$\longrightarrow 0 \ \ as \ \ \delta_1, \delta_2 \to 0.$$

PROOF. We proceed using several Taylor series expansions to evaluate the conditional expectation above. Note that

$$J(\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)) = J(\theta(n) + \delta_1\triangle(n)) + \delta_2 \sum_{k=1}^{N} \hat{\triangle}_k(n)\nabla_k J(\theta(n) + \delta_1\triangle(n))$$

$$+\frac{1}{2}\delta_2^2 \sum_{k=1}^{N}\sum_{l=1}^{N} \hat{\triangle}_k(n)\nabla^2_{k,l} J(\theta(n) + \delta_1\triangle(n))\hat{\triangle}_l(n) + o\big(\delta_2^2\big).$$

Similarly,

$$J(\theta(n) - \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)) = J(\theta(n) - \delta_1\triangle(n)) + \delta_2 \sum_{k=1}^{N} \hat{\triangle}_k(n)\nabla_k J(\theta(n) - \delta_1\triangle(n))$$

$$+\frac{1}{2}\delta_2^2 \sum_{k=1}^{N}\sum_{l=1}^{N} \hat{\triangle}_k(n)\nabla^2_{k,l} J(\theta(n) - \delta_1\triangle(n))\hat{\triangle}_l(n) + o\big(\delta_2^2\big).$$

After some rearrangement of terms, it is easy to see that

$$E\left[\left(\left(\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))-J(\theta(n)+\delta_1\triangle(n))}{\delta_2\hat{\triangle}_j(n)}\right)\right.\right.$$

$$\left.\left.-\left(\frac{J(\theta(n)-\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))-J(\theta(n)-\delta_1\triangle(n))}{\delta_2\hat{\triangle}_j(n)}\right)\right)\Big/2\delta_1\triangle_i(n)\mid\mathcal{F}(n)\right]$$

$$=E\left[\frac{\nabla_j J(\theta(n)+\delta_1\triangle(n))-\nabla_j J(\theta(n)-\delta_1\triangle(n))}{2\delta_1\triangle_i(n)}\right.$$

$$+\sum_{k\neq j}\frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)}\frac{\nabla_k J(\theta(n)+\delta_1\triangle(n))-\nabla_k J(\theta(n)-\delta_1\triangle(n))}{2\delta_1\triangle_i(n)}$$

$$+\delta_2\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\hat{\triangle}_k(n)(\nabla^2_{k,l}J(\theta(n)+\delta_1\triangle(n))-\nabla^2_{k,l}J(\theta(n)-\delta_1\triangle(n)))\hat{\triangle}_l(n)}{4\delta_1\triangle_i(n)\hat{\triangle}_j(n)}$$

$$\left.+o(\delta_2)\mid\mathcal{F}(n)\right]. \tag{42}$$

Now using Taylor series expansions of $\nabla_j J(\theta(n)+\delta_1\triangle(n))$ and $\nabla_j J(\theta(n)-\delta_1\triangle(n))$ around $\nabla_j J(\theta(n))$ gives

$$\frac{\nabla_j J(\theta(n)+\delta_1\triangle(n))-\nabla_j J(\theta(n)-\delta_1\triangle(n))}{2\delta_1\triangle_i(n)}=\nabla^2_{j,i}J(\theta(n))$$

$$+\sum_{l\neq i}\frac{\triangle_l(n)}{\triangle_i(n)}\nabla^2_{j,l}J(\theta(n))+o(\delta_1^2).$$

A similar expansion can be obtained with index k in place of j in the second term on the RHS of (42). Also note that

$$\frac{\nabla^2_{k,l}J(\theta(n)+\delta_1\triangle(n))-\nabla^2_{k,l}J(\theta(n)-\delta_1\triangle(n))}{4\delta_1\triangle_i(n)}=\sum_{m=1}^{N}\frac{\triangle_m(n)\nabla^3_{k,l,m}J(\theta(n))}{2\triangle_i(n)}+o(\delta_1)$$

Thus,

$$\delta_2\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\hat{\triangle}_k(n)(\nabla^2_{k,l}J(\theta(n)+\delta_1\triangle(n))-\nabla^2_{k,l}J(\theta(n)-\delta_1\triangle(n)))\hat{\triangle}_l(n)}{4\delta_1\triangle_i(n)\hat{\triangle}_j(n)}$$

$$=\delta_2\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{m=1}^{N}\frac{\hat{\triangle}_k(n)\triangle_m(n)\nabla^3_{k,l,m}J(\theta(n))\hat{\triangle}_l(n)}{2\hat{\triangle}_j(n)\triangle_i(n)}+o(\delta_1).$$

Substituting the above in (42), one obtains

$$E\left[\left(\left(\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))-J(\theta(n)+\delta_1\triangle(n))}{\delta_2\hat{\triangle}_j(n)}\right)\right.\right.$$

$$-\left(\frac{J(\theta(n)-\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))-J(\theta(n)-\delta_1\triangle(n))}{\delta_2\hat{\triangle}_j(n)}\right)\Bigg)\Bigg/2\delta_1\triangle_i(n)\ \mid\ \mathcal{F}(n)\Bigg]$$

$$= E\Bigg[\nabla^2_{j,i}J(\theta(n))+\sum_{l\neq i}\frac{\triangle_l(n)}{\triangle_i(n)}\nabla^2_{j,l}J(\theta(n))+\sum_{k\neq j}\frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)}\nabla^2_{k,i}J(\theta(n))$$

$$+\sum_{k\neq j}\sum_{l\neq i}\frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)}\frac{\triangle_l(n)}{\triangle_i(n)}\nabla^2_{k,l}J(\theta(n))+\delta_2\sum_{k,l,m=1}^{N}\frac{\hat{\triangle}_k(n)\triangle_m(n)\nabla^3_{k,l,m}J(\theta(n))\hat{\triangle}_l(n)}{2\hat{\triangle}_j(n)\triangle_i(n)}$$

$$+o(\delta_1)+o(\delta_2)\ \mid\ \mathcal{F}(n)\Bigg]$$

$$=\nabla^2_{j,i}J(\theta(n))+\sum_{l\neq i}E\left[\frac{\triangle_l(n)}{\triangle_i(n)}\ \mid\ \mathcal{F}(n)\right]\nabla^2_{j,l}J(\theta(n))$$

$$+\sum_{k\neq j}E\left[\frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)}\ \mid\ \mathcal{F}(n)\right]\nabla^2_{k,i}J(\theta(n))$$

$$+\sum_{k\neq j}\sum_{l\neq i}E\left[\frac{\hat{\triangle}_k(n)}{\hat{\triangle}_j(n)}\frac{\triangle_l(n)}{\triangle_i(n)}\ \mid\ \mathcal{F}(n)\right]\nabla^2_{k,l}J(\theta(n))$$

$$+\delta_2\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{m=1}^{N}E\left[\frac{\hat{\triangle}_k(n)\hat{\triangle}_l(n)\triangle_m(n)}{2\hat{\triangle}_j(n)\triangle_i(n)}\ \mid\ \mathcal{F}(n)\right]\nabla^3_{k,l,m}J(\theta(n))+o(\delta_1)+o(\delta_2).$$

Now by conditions on $\triangle_i(n)$, $\hat{\triangle}_i(n)$, $n\geq 1$, $i=1,\ldots,N$ (Assumption (B)), it is easy to see that all conditional expectations in the last equality above equal zero. Thus

$$E\Bigg[\Bigg(\left(\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))-J(\theta(n)+\delta_1\triangle(n))}{\delta_2\hat{\triangle}_j(n)}\right)$$

$$-\left(\frac{J(\theta(n)-\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))-J(\theta(n)-\delta_1\triangle(n))}{\delta_2\hat{\triangle}_j(n)}\right)\Bigg)\Bigg/2\delta_1\triangle_i(n)\ \mid\ \mathcal{F}(n)\Bigg]$$

$$=\nabla^2_{j,i}J(\theta(n))+o(\delta_1)+o(\delta_2).$$

The claim follows. □

Consider now the following ODEs: For $j,i=1,\ldots,N$,

$$\dot{H}_{j,i}(t)=\nabla^2_{j,i}J(\theta(t))-H_{j,i}(t),$$

$$\dot{\theta}_i(t)=0. \tag{43}$$

Next, define $\{r(n)\}$ as follows: $r(0) = 0$ and for $n > 0$, $r(n) = \sum_{m=0}^{n-1} c(m)$. Define $\bar{H}(t) = [[\bar{H}_{j,i}(t)]]_{j,i=1}^{N}$ and $\bar{x}^{w}(t)$, $w \in \{-, +, -+, ++\}$ as follows: for $j, i = 1, \ldots, N$, $\bar{H}_{j,i}(r(n)) = H_{j,i}(n)$, and $\bar{x}^{w}(r(n)) = Z^{w}(nL)$ with linear interpolations on $[r(n), r(n+1)]$. We now have

Lemma A.8. *Given $T, \gamma > 0$, $(\theta(r(n) + \cdot), \bar{H}(r(n) + \cdot))$ is a bounded (T, γ)-perturbation of* (43) *for sufficiently large n.*

Proof. Note that recursion (15) in the algorithm can be rewritten as: for $j, i = 1, \ldots, N$,

$$\begin{aligned} H_{j,i}(n+1) \;=\; & H_{j,i}(n) + c(n)\big(\nabla^2_{j,i} J(\theta(n))\big) + \hat{\zeta}(n) + \zeta(n) - H_{j,i}(n)) \\ & + (N_{j,i}(n+1) - N_{j,i}(n)), \end{aligned}$$

with appropriately defined $\hat{\zeta}(n)$ and $\zeta(n)$ that are, however, both $o(1)$ by Proposition A.7 and Corollary A.5. Also by the foregoing, $(N_{j,i}(n+1) - N_{j,i}(n))$ is $o(1)$. Next, rewrite (16) as follows: for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Pi_i(\theta_i(n) + c(n)\beta(n)),$$

where $\beta(n) = o(1)$ since $a(n) = o(c(n))$. The claim follows. □

Suppose $H(n) = [[H_{j,i}(n)]]_{j,i=1}^{N}$. It is now easy to see as in Corollary A.5 that:

Lemma A.9. *$\| H(n) - \nabla^2 J(\theta(n)) \| \to 0$ a.s. as $\delta_1, \delta_2 \to 0$ and $n \to \infty$.*

Next, we have

Corollary A.10. *With probability one, $\| \{\Gamma(H(n))\}^{-1} - \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1} \| \to 0$ as $\delta_1, \delta_2 \to 0$ and $n \to \infty$.*

Proof. Note that

$$\| \{\Gamma(H(n))\}^{-1} - \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1} \|$$

$$= \| \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1}(\Gamma(\nabla^2 J(\theta(n)))\{\Gamma(H(n))\}^{-1} - I) \|$$

$$= \| \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1}(\Gamma(\nabla^2 J(\theta(n)))\{\Gamma(H(n))\}^{-1} - \Gamma(H(n))\{\Gamma(H(n))\}^{-1}) \|$$

$$= \| \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1}(\Gamma(\nabla^2 J(\theta(n))) - \Gamma(H(n)))\{\Gamma(H(n))\}^{-1} \|$$

$$\leq \| \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1} \| \cdot \| \Gamma(\nabla^2 J(\theta(n))) - \Gamma(H(n)) \| \cdot \| \{\Gamma(H(n))\}^{-1} \|$$

$$\leq \sup_n \| \{\Gamma(\nabla^2 J(\theta(n)))\}^{-1} \| \sup_n \| \{\Gamma(H(n))\}^{-1} \| \cdot \| \Gamma(\nabla^2 J(\theta(n))) - \Gamma(H(n)) \|$$

$$\to 0 \ \text{ as } \ n \to \infty,$$

by Assumption (D). I denotes the $N \times N$-identity matrix. The first inequality follows from the property on induced matrix norms (see Proposition A.12 of Bertsekas and Tsitsiklis [1989]). The claim follows. □

Proof of Theorem 3.1. For $i = 1, \ldots, N$, let $\{R_i(n),\ n \geq 1\}$ be defined according to

$$R_i(n) = \sum_{m=0}^{n-1} a(m) \sum_{k=1}^{N} \bar{M}_{i,k}(\theta(m)) \left(\frac{J(\theta(m) - \delta_1 \triangle(m)) - J(\theta(m) + \delta_1 \triangle(m))}{2\delta_1 \triangle_k(m)} \right.$$

$$\left. - E \left[\frac{J(\theta(m) - \delta_1 \triangle(m)) - J(\theta(m) + \delta_1 \triangle(m))}{2\delta_1 \triangle_k(m)} \ \mid\ \mathcal{F}(m) \right] \right),$$

$n \geq 1$. Then it is easy to check that $\{R_i(n), \mathcal{F}(n)\}$, $i = 1, \ldots, N$, form martingale sequences that are almost surely convergent. Now recursion (16) of the algorithm can be rewritten as

$$\theta_i(n+1) = \Pi_i \left(\theta_i(n) + a(n) \sum_{k=1}^{N} \bar{M}_{i,k}(\theta(n))\ E\left[(J(\theta(n) - \delta_1 \triangle(n)) \right.\right.$$

$$\left.\left. -\ J(\theta(n) + \delta_1 \triangle(n)))/2\delta_1 \triangle_k(n)\ \mid\ \mathcal{F}(n)\right] + (R_i(n+1) - R_i(n)) + a(n)\alpha(n) \right), \tag{44}$$

where $(R_i(n+1) - R_i(n))$ is $o(1)$ and $\alpha(n)$ vanishes as $n \to \infty$ and $\delta_1, \delta_2 \to 0$ by Corollaries A.5 and A.10. Further, using Taylor series expansions of $J(\theta(n) -\delta_1 \triangle(n))$ and $J(\theta(n) +\delta_1 \triangle(n))$, respectively, around $\theta(n)$ and taking the conditional expectation, it is easy to see that recursion (16) can now be rewritten as

$$\theta_i(n+1) = \Pi_i \left(\theta_i(n) - a(n) \sum_{k=1}^{N} \bar{M}_{i,k}(\theta(n)) \nabla_k J(\theta(n)) + a(n)\xi_{\delta_1}(n) \right.$$

$$\left. + (R_i(n+1) - R_i(n)) + a(n)\alpha(n) \right), \tag{45}$$

where $\xi_{\delta_1}(n)$ vanishes as $n \to \infty$ and $\delta_1 \to 0$. Note that (45) can be viewed, using a standard approximation argument as on pages 191–194 of Kushner and Clark [1978] and Proposition A.7, as a discretization of the ODE (17) with certain error terms that, however, vanish asymptotically (as $n \to \infty$) and in the limit as $\delta_1, \delta_2 \to 0$. Now (17) can be written as (see Eq. (3.1) of Kushner and Yin [1997])

$$\dot{\theta} = -\bar{M}(\theta)\nabla J(\theta) + z, \quad z(t) \in -\bar{C}(\theta(t)), \tag{46}$$

where $z(\cdot)$ is the projection term. For $\theta \in C^o$, $\bar{C}(\theta)$ contains only the zero element, and for $\theta \in \partial C$ (boundary of C), $\bar{C}(\theta)$ is the infinite convex cone generated by the outer normals at θ of the faces on which θ lies. Note that $J(\theta)$ itself serves as an associated Liapunov function for (46) since (Kushner and Yin [1997] pp. 75)

$$\frac{dJ(\theta)}{dt} = \nabla J(\theta)^T \dot{\theta} = \nabla J(\theta)^T(-\bar{M}(\theta)\nabla J(\theta) + z) \leq 0.$$

In particular, for $\theta \in \hat{K}$ (i.e., $z = 0$), $\frac{dJ(\theta)}{dt} < 0$ if $\nabla J(\theta) \neq 0$. Now $J(\theta)$ is uniformly bounded since the cost function $h(\cdot)$ is bounded. Let $\lambda = \sup_\theta J(\theta) < \infty$. Then

$\{\theta \mid J(\theta) \leq \lambda\} = C$. It follows from Lasalle's invariance theorem [Lasalle and Lefschetz 1961] stated also as Theorem 2.3, on page 76 of Kushner and Yin [1997] that $\theta(t) \to \{\theta \in C \mid \nabla J(\theta)^T\ \tilde{\pi}(-\bar{M}(\theta)\nabla J(\theta)) = 0\}$ as $t \to \infty$. The claim follows. □

A.2 Algorithm 3SA

It can be shown as in Corollary A.5 that almost surely,

$$\| Z^w(nL) - J(\theta^w(n)) \| \to 0 \ \ \forall w \in \{-, +, ++\}.$$

Now define $\mathcal{F}_1(n)$, $n \geq 1$ by $\mathcal{F}_1(n) = \sigma(\theta_i(m), H_{j,i}(m), Z^-(mL), Z^+(mL), Z^{++}(mL), m \leq n, i, j = 1, \ldots, N; \triangle(m), \hat{\triangle}(m), m < n)$.

Proposition A.11. *With probability one,* $\forall j, i \in \{1, \ldots, N\}$

$$\left| E\left[\frac{J(\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)) - J(\theta(n) + \delta_1\triangle(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} \ \middle|\ \mathcal{F}_1(n)\right] - \nabla^2_{j,i}J(\theta(n))\right|$$

$$\longrightarrow 0 \ \ as \ \ \delta_1, \delta_2 \to 0.$$

Proof. Note as before that

$$\frac{J(\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)) - J(\theta(n) + \delta_1\triangle(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} = \sum_{l=1}^{N} \frac{\hat{\triangle}_l(n)\nabla_l J(\theta(n) + \delta_1\triangle(n))}{\delta_1\triangle_i(n)\hat{\triangle}_j(n)}$$

$$+\frac{\delta_2}{2}\sum_{l=1}^{N}\sum_{m=1}^{N} \frac{\hat{\triangle}_l(n)\hat{\triangle}_m(n)\nabla^2_{l,m}J(\theta(n) + \delta_1\triangle(n))}{\delta_1\triangle_i(n)\hat{\triangle}_j(n)} + o(\delta_2) \tag{47}$$

Taking again appropriate Taylor series expansions of $\nabla_l J(\theta(n) + \delta_1\triangle(n))$ and $\nabla^2_{l,m}J(\theta(n) + \delta_1\triangle(n))$ around $\theta(n)$, substituting in (47), and taking conditional expectation with respect to $\mathcal{F}_1(n)$, one obtains

$$E\left[\frac{J(\theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)) - J(\theta(n) + \delta_1\triangle(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} \ \middle|\ \mathcal{F}_1(n)\right]$$

$$= \sum_{l=1}^{N} \frac{1}{\delta_1} E\left[\frac{\hat{\triangle}_l(n)}{\triangle_i(n)\hat{\triangle}_j(n)} \ \middle|\ \mathcal{F}_1(n)\right] \nabla_l J(\theta(n)) + \nabla^2_{j,i}J(\theta(n))$$

$$+ \sum_{l=1, l\neq j}^{N} \ \sum_{k=1, k\neq i}^{N} E\left[\frac{\hat{\triangle}_l(n)\triangle_k(n)}{\hat{\triangle}_j(n)\triangle_i(n)} \ \middle|\ \mathcal{F}_1(n)\right] \nabla^2_{l,k}J(\theta(n))$$

$$+\frac{\delta_1}{2}\sum_{l=1}^{N}\sum_{k=1}^{N}\sum_{m=1}^{N} E\left[\frac{\hat{\triangle}_l(n)\triangle_k(n)\triangle_m(n)}{\triangle_i(n)\hat{\triangle}_j(n)} \ \middle|\ \mathcal{F}_1(n)\right] \nabla^3_{k,m,l}J(\theta(n))$$

$$+\frac{\delta_2}{2\delta_1}\sum_{l=1}^{N}\sum_{m=1}^{N} E\left[\frac{\hat{\triangle}_l(n)\hat{\triangle}_m(n)}{\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_1(n)\right]\nabla^2_{l,m}J(\theta(n))$$

$$+\frac{\delta_2}{2}\sum_{l=1}^{N}\sum_{m=1}^{N}\sum_{k=1}^{N} E\left[\frac{\hat{\triangle}_l(n)\hat{\triangle}_m(n)\triangle_k(n)}{\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_1(n)\right]\nabla^3_{l,m,k}J(\theta(n)) + o(\delta_1) + o(\delta_2).$$

It is easy to see from Assumption (B) as before that all the conditional expectation terms on the RHS above equal zero. Thus

$$E\left[\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n)) - J(\theta(n)+\delta_1\triangle(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_1(n)\right]$$

$$= \nabla^2_{j,i}J(\theta(n)) + o(\delta_1) + o(\delta_2).$$

The claim follows. □

The proof of Theorem 3.2 now proceeds along exactly similar lines as that of Theorem 3.1.

A.3 Algorithm 2SA

One can show as in Corollary A.5 that with probability one,

$$\| Z^w(nL) - J(\theta^w(n)) \| \to 0 \quad \forall w \in \{+, ++\}.$$

Now define σ-fields $\mathcal{F}_2(n), n \geq 1$ by $\mathcal{F}_2(n) = \sigma(\theta_i(m), H_{j,i}(m), Z^+(mL), Z^{++}(mL), m \leq n, i, j = 1, \ldots, N;\ \triangle(m), \hat{\triangle}(m), m < n)$. Since the form of Hessian estimate here is exactly the same as in Algorithm 3SA, the conclusions of Proposition A.11 continue to hold with $\mathcal{F}_2(n)$ in place of $\mathcal{F}_1(n)$. We have

Proof of Theorem 3.3. Using an appropriate martingale construction as before, it is easy to see that recursion (26) of Algorithm 2SA can be rewritten as in (44) as

$$\theta_i(n+1) = \Pi_i\left(\theta_i(n) + a(n)\sum_{k=1}^{N}\bar{M}_{i,k}(\theta(n))(E[(J(\theta(n)+\delta_1\triangle(n)) \right.$$

$$\left. - J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n)))/\delta_2\hat{\triangle}_k(n) \mid \mathcal{F}_2(n)]) + \alpha_1(n) + a(n)\alpha_2(n)\right), \quad (48)$$

where $\alpha_1(n)$ is $o(1)$ and $\alpha_2(n)$ becomes asymptotically negligible as $\delta_1, \delta_2 \to 0$. We now use appropriate Taylor series expansions in the second term on the RHS of (48). Note that

$$\frac{J(\theta(n)+\delta_1\triangle(n)) - J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))}{\delta_2\hat{\triangle}_k(n)} = -\sum_{l=1}^{N}\frac{\hat{\triangle}_l(n)}{\hat{\triangle}_k(n)}\nabla_l J(\theta(n)+\delta_1\triangle(n))$$

$$-\frac{\delta_2}{2}\sum_{l=1}^{N}\sum_{j=1}^{N}\frac{\hat{\triangle}_l(n)}{\hat{\triangle}_k(n)}\hat{\triangle}_j(n)\nabla^2_{l,j}J(\theta(n)+\delta_1\triangle(n)) + o(\delta_2). \quad (49)$$

Again,

$$\nabla_l J(\theta(n)+\delta_1\triangle(n)) = \nabla_l J(\theta(n)) + \delta_1 \sum_{j=1}^{N} \triangle_j(n)\nabla^2_{l,j} J(\theta(n)) + o(\delta_1),$$

$$\nabla^2_{l,j} J(\theta(n)+\delta_1\triangle(n)) = \nabla^2_{l,j} J(\theta(n)) + \delta_1 \sum_{m=1}^{N} \triangle_m(n)\nabla^3_{l,j,m} J(\theta(n)) + o(\delta_1).$$

Substituting the above in (49) and taking conditional expectations, we have

$$E\left[\frac{J(\theta(n)+\delta_1\triangle(n)) - J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))}{\delta_2\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_2(n)\right]$$

$$= -\nabla_k J(\theta(n)) - \sum_{l=1,l\neq k}^{N} E\left[\frac{\hat{\triangle}_l(n)}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_2(n)\right] \nabla_l J(\theta(n))$$

$$-\delta_1 \sum_{l=1}^{N}\sum_{j=1}^{N} E\left[\frac{\hat{\triangle}_l(n)\triangle_j(n)}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_2(n)\right] \nabla^2_{l,j} J(\theta(n))$$

$$-\frac{\delta_2}{2} \sum_{l=1}^{N}\sum_{j=1}^{N} E\left[\frac{\hat{\triangle}_l(n)\hat{\triangle}_j(n)}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_2(n)\right] \nabla^2_{l,j} J(\theta(n))$$

$$-\frac{\delta_1\delta_2}{2} \sum_{l=1}^{N}\sum_{j=1}^{N}\sum_{m=1}^{N} E\left[\frac{\hat{\triangle}_l(n)\hat{\triangle}_j(n)\triangle_m(n)}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_2(n)\right] \nabla^3_{l,j,m} J(\theta(n)) + o(\delta_1) + o(\delta_2).$$

Now it is easy to see using Assumption (B) that all conditional expectation terms on the RHS above equal zero. Thus,

$$E\left[\frac{J(\theta(n)+\delta_1\triangle(n)) - J(\theta(n)+\delta_1\triangle(n)+\delta_2\triangle(n))}{\delta_2\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_2(n)\right]$$

$$= -\nabla_k J(\theta(n)) + o(\delta_1) + o(\delta_2).$$

The rest now follows in exactly the same manner as in Theorem 3.1. □

A.4 Algorithm (1SA)

We show in the following that the bias terms in both the gradient and Hessian vanish asymptotically in the mean. One can show as in Corollary A.5 that almost surely,

$$\| Z^{++}(nL) - J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n)) \| \rightarrow 0,$$

as $n \rightarrow \infty$. Define $\mathcal{F}_3(n)$, $n \geq 1$ by $\mathcal{F}_3(n) = \sigma(\theta_i(m), H_{j,i}(m), Z^{++}(mL), m \leq n, i, j = 1, \ldots, N; \triangle(m), \hat{\triangle}(m), m < n)$. We then have

PROPOSITION A.12. *With probability one, $\forall j, i \in \{1, \ldots, N\}$,*

$$\lim_{\delta_1,\delta_2\to 0}\left| E\left[\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_3(n)\right] - \nabla^2_{j,i}J(\theta(n))\right| = 0. \tag{50}$$

PROOF. Note that the proof here is similar to that of Proposition A.11, the only difference being the presence of additional bias terms that arise from the Taylor series expansion of the 'extra' term $E[\frac{J(\theta(n)+\delta_1\triangle(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} \mid \mathcal{F}_3(n)]$ that, in turn, results from the Taylor's expansion of the first term in (50). The above extra term does not contribute to the bias in Algorithm 2SA because it is contained there in the conditional average of the Hessian estimate itself. Now note that

$$E\left[\frac{J(\theta(n)+\delta_1\triangle(n))}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_3(n)\right] = E\left[\frac{1}{\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_3(n)\right]\frac{J(\theta(n))}{\delta_1\delta_2}$$

$$+\sum_{k=1}^{N} E\left[\frac{\triangle_k(n)}{\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_3(n)\right]\frac{\nabla_k J(\theta(n))}{\delta_2}$$

$$+\frac{\delta_1}{2\delta_2}\sum_{k=1}^{N}\sum_{m=1}^{N} E\left[\frac{\triangle_k(n)\triangle_m(n)}{\triangle_i(n)\hat{\triangle}_j(n)} \;\middle|\; \mathcal{F}_3(n)\right]\nabla^2_{k,m}J(\theta(n)) + o(\delta_1).$$

It is easy to see from Assumption (B) that all conditional expectation terms on the RHS above equal zero. The rest follows as in Proposition A.11. □

PROOF OF THEOREM 3.4. Note that as earlier, (29) can be rewritten using a martingale argument as

$$\theta_i(n+1) = \Pi_i(\theta_i(n) - a(n)\sum_{k=1}^{N}\bar{M}_{i,k}(\theta(n))E\left[\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))}{\delta_2\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_3(n)\right]$$

$$+\alpha_3(n) + a(n)\alpha_4(n)),$$

where $\alpha_3(n)$ is $o(1)$ and $\alpha_4(n)$ vanishes asymptotically as $\delta_1, \delta_2 \to 0$. Now observe that

$$\frac{J(\theta(n)+\delta_1\triangle(n)+\delta_2\hat{\triangle}(n))}{\delta_2\hat{\triangle}_k(n)} = \frac{J(\theta(n)+\delta_1\triangle(n))}{\delta_2\hat{\triangle}_k(n)} + \sum_{l=1}^{N}\frac{\hat{\triangle}_l(n)}{\hat{\triangle}_k(n)}\nabla_l J(\theta(n)+\delta_1\triangle(n))$$

$$+\frac{\delta_2}{2}\sum_{l=1}^{N}\sum_{j=1}^{N}\frac{\hat{\triangle}_l(n)}{\hat{\triangle}_k(n)}\hat{\triangle}_j(n)\nabla^2_{l,j}J(\theta(n)+\delta_1\triangle(n)) + o(\delta_2). \tag{51}$$

Upon comparison with (49), it is clear that there is an extra term $\frac{J(\theta(n)+\delta_1\triangle(n))}{\delta_2\hat{\triangle}(n)}$ on the RHS of (51) that is not present in the corresponding expression in (49). Again note that

$$E\left[\frac{J(\theta(n)+\delta_1\triangle(n))}{\delta_2\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_3(n)\right] = E\left[\frac{1}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_3(n)\right]\frac{J(\theta(n))}{\delta_2}$$

$$+\delta_1 \sum_{l=1}^{N} E\left[\frac{\triangle_l(n)}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_3(n)\right] \frac{\nabla_l J(\theta(n))}{\delta_2}$$

$$+\frac{\delta_1^2}{2} \sum_{l=1}^{N} \sum_{m=1}^{N} E\left[\frac{\triangle_l(n)\triangle_m(n)}{\hat{\triangle}_k(n)} \;\middle|\; \mathcal{F}_3(n)\right] \frac{\nabla^2_{l,m} J(\theta(n))}{\delta_2} + o(\delta_1).$$

It is easy to see from Assumption (B) that all conditional expectation terms on the RHS above equal zero. The rest now follows as in Theorem 3.3. □

ACKNOWLEDGMENTS

The author thanks Area Editor Prof. M. K. Nakayama, the Associate Editor and both the referees for their many comments which helped in enhancing the scope of the experiments and improving the quality of the manuscript.

REFERENCES

Andradóttir, S. 1996. Optimization of the transient and steady-state behavior of discrete event systems. *Manag. Sci. 42*, 5, 717–737.

Bertsekas, D. P. 1999. *Nonlinear Programming*. Athena Scientific, Belmont.

Bertsekas, D. P. and Tsitsiklis, J. N. 1989. *Parallel and Distributed Computation*. Prentice Hall, New Jersey.

Bhatnagar, S. 1997. *Multiscale Stochastic Approximation Algorithms with Applications to ABR Service in ATM Networks*. Ph. D. thesis, 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. and Info. Sci. 11*, 509–522.

Bhatnagar, S. and Borkar, V. S. 1998. A two time scale stochastic approximation scheme for simulation based parametric optimization. *Prob. Eng. and Info. Sci. 12*, 519–531.

Bhatnagar, S. and Borkar, V. S. 2003. Multiscale chaotic SPSA and smoothed functional algorithms for simulation optimization. *Simulation 79*, 10, 568–580.

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. Network. 9*, 4, 479–491.

Bhatnagar, S., Fu, M. C., Marcus, S. I., and Wang, I.-J. 2003. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Trans. Modell. Comput. Simul. 13*, 2, 180–209.

Brandiere, O. 1998. Some pathological traps for stochastic approximation. *SIAM J. Contr. Optim. 36*, 1293–1314.

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.

Dippon, J. and Renz, J. 1997. Weighted means in stochastic approximation of minima. *SIAM J. Contr. Optim. 35*, 1811–1827.

Fabian, V. 1971. Stochastic approximation. In *Optimizing Methods in Statistics* J. J. Rustagi, Ed. Academic Press, New York, NY, 439–470.

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.

GELFAND, S. B. AND MITTER, S. K. 1991. Recursive stochastic algorithms for global optimization in $\mathcal{R}^{d*}$. *SIAM J. Cont. Optim. 29*, 5, 999–1018.

HIRSCH, M. W. 1989. Convergent activation dynamics in continuous time networks. *Neural Networks 2*, 331–349.

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

KIEFER, E. AND WOLFOWITZ, J. 1952. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist. 23*, 462–466.

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

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

KUSHNER, H. J. AND YIN, G. G. 1997. *Stochastic Approximation Algorithms and Applications*. Springer Verlag, New York, NY.

LASALLE, J. P. AND LEFSCHETZ, S. 1961. *Stability by Liapunov's Direct Method with Applications*. Academic Press, New York, NY.

L'ECUYER, P. AND GLYNN, P. W. 1994. Stochastic optimization by simulation: convergence proofs for the $GI/G/1$ queue in steady-state. *Manag. Sci. 40*, 11, 1562–1578.

LUMAN, R. R. 2000. Upgrading complex systems of systems: a CAIV methodology for warfare area requirements allocation. *Military Operations Research 5*, 2, 53–75.

PEMANTLE, R. 1990. Nonconvergence to unstable points in urn models and stochastic approximations. *Annals of Prob. 18*, 698–712.

POLYAK, B. T. AND JUDITSKY, A. B. 1992. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim. 30*, 4, 838–855.

ROBBINS, H. AND MONRO, S. 1951. A stochastic approximation method. *Ann. Math. Statist. 22*, 400–407.

RUPPERT, D. 1985. A Newton-Raphson version of the multivariate Robbins-Monro procedure. *Annals Statist. 13*, 236–245.

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. 2000. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Trans. Autom. Contr. 45*, 1839–1853.

ZHU, X. AND SPALL, J. C. 2002. A modified second-order SPSA optimization algorithm for finite samples. *Int. J. Adapt. Contr. Sign. Proce. 16*, 397–409.

Received October 2003; revised May 2004; accepted October 2004