Adaptive Newton-Based Multivariate Smoothed Functional Algorithms for Simulation Optimization

SHALABH BHATNAGAR
Indian Institute of Science, Bangalore

In this article, we present three smoothed functional (SF) algorithms for simulation optimization. While one of these estimates only the gradient by using a finite difference approximation with two parallel simulations, the other two are adaptive Newton-based stochastic approximation algorithms that estimate both the gradient and Hessian. One of the Newton-based algorithms uses only one simulation and has a one-sided estimate in both the gradient and Hessian, while the other uses two-sided estimates in both quantities and requires two simulations. For obtaining gradient and Hessian estimates, we perturb each parameter component randomly using independent and identically distributed (i.i.d) Gaussian random variates.

The earlier SF algorithms in the literature only estimate the gradient of the objective function. Using similar techniques, we derive two unbiased SF-based estimators for the Hessian and develop suitable three-timescale stochastic approximation procedures for simulation optimization. We present a detailed convergence analysis of our algorithms and show numerical experiments with parameters of dimension 50 on a setting involving a network of $M/G/1$ queues with feedback. We compare the performance of our algorithms with related algorithms in the literature. While our two-simulation Newton-based algorithm shows the best results overall, our one-simulation algorithm shows better performance compared to other one-simulation algorithms.

Categories and Subject Descriptors: I.6.1 [**Simulation and Modeling**]: Simulation Theory; G.3 [**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: Smoothed functional algorithms, three-timescale stochastic approximation, simulation optimization, Gaussian perturbations, Newton-based algorithms

ACM Reference Format:

Bhatnagar, S. 2007. Adaptive Newton-based multivariate smoothed functional algorithms for simulation optimization. ACM Trans. Model. Comput. Simul. 18, 1, Article 2 (December 2007), 35 pages. DOI = 10.1145/1315575.1315577 http://doi.acm.org/10.1145/1315575.1315577

This work was supported in part by Grants SR/S3/EE/43/2002-SERC-Engg and SR/S3/EECE/011/2007 from the Department of Science and Technology, Goverment of India.

Author's address: Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India; email: shalabh@csa.iisc.ernet.in.

1. INTRODUCTION

The Robbins-Monro stochastic approximation algorithm (see Robbins and Monro [1951]) is used to find zeros of an objective function whose noisy estimates are available either as outputs of a real system or via simulation. In the case of gradient search algorithms, the objective function whose zeros are to be found is typically the gradient of average cost. Most often, the average cost itself does not possess an analytic expression; hence obtaining its gradient through direct methods is not possible. Simulation has been widely used in such scenarios. Techniques based on perturbation analysis (PA) (Ho and Cao [1991]), and likelihood ratio (LR) (L'Ecuyer and Glynn [1994]), typically require a single simulation run per gradient estimate (we call such algorithms *one-simulation* methods). However, they require certain constraining assumptions on the underlying process parameters and sample performance. LR-based approaches also rely on a change of measure with respect to which the average cost expectation is taken. It is assumed in PA- and LR-based approaches that the gradient of sample performance exists. Further, in many of these approaches, the parameter is updated only at certain regeneration epochs which can be sparse in large or high-dimensional systems.

In optimization approaches based on estimating gradients, the two-sided (one-sided) Kiefer-Wolfowitz (K-W) gradient estimates (cf. Kiefer and Wolfowitz [1952]) require $2N$ $(N + 1)$ parallel simulations of the system to estimate an N-dimensional parameter. Spall's [1992] simultaneous perturbation stochastic approximation (SPSA) algorithm, on the other hand, requires only two parallel simulations by using random perturbations for all parameter components in the gradient estimate and has been found to perform well in various settings. A related one-simulation algorithm of Spall [1997], however, has not been found to perform as well because of certain "additional" bias terms in the gradient estimate. In the smoothed functional (SF) scheme originally due to Katkovnik and Kulchitsky (see Katkovnik and Kulchitsky [1972] and Rubinstein [1981]), the idea is to approximate the gradient of expected performance by its convolution with a multivariate normal distribution. In Styblinski and Opalski [1986], the SF scheme is applied for yield gradient estimation to optimize certain parameters of the integrated circuit (IC) manufacturing process. While the original SF algorithm in Katkovnik and Kulchitsky [1972] uses only one simulation, in Styblinski and Tang [1990] and Chin [1997], a related two-simulation SF algorithm based on a finite difference gradient estimate was presented. The latter algorithm has been shown (Styblinski and Tang [1990]) to have less variability compared with the one-simulation SF algorithm (Katkovnik and Kulchitsky [1972]; Styblinski and Opalski [1986]).

There is also considerable work on adaptive Newton-based schemes that estimate the Hessian in addition to 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], the objective function gradients were assumed known and the Hessian was estimated using finite gradient differences. In Spall [2000], a Newton-type stochastic adaptive algorithm has

been proposed. The Hessian estimates there are obtained using four objective function samples in cases where the gradients are not known and three samples in cases where these are known. The estimates in Spall [2000] were based on the simultaneous perturbation methodology. These (Hessian) estimates at each update epoch are projected onto the set of positive definite and symmetric matrices so that the algorithm performs a search in the *descent direction* in order to converge to a minimum. Certain mappings for obtaining the above projected Hessian estimates were described in Bertsekas [1999] and Spall [2000]. In Zhu and Spall [2002], another mapping based on projecting the eigenvalues of Hessian updates onto the positive half line was given. The algorithm there replaces the inverse of the projected Hessian update with that of the geometric mean of the projected eigenvalues in the parameter recursion step. In Spall [2006], a variant of the adaptive algorithm of Spall [2000] has recently been proposed.

Two-timescale stochastic approximation algorithms have been used for finding solutions to systems of equations involving two nested loops. For instance, gradient search-based algorithms for simulation optimization that use two timescales have been developed in Bhatnagar and Borkar [1998, 2003], Bhatnagar et al. [2001, 2003]. Since the aim in these is to find the gradient of average cost, one requires that the cost function corresponding to any given parameter update be averaged before the next parameter update step. Using two-timescale stochastic approximation, both the above steps are allowed to move in tandem, one after the other. Thus cost is averaged on the faster timescale recursion while the slower recursion performs a gradient search. The above schemes are efficient alternatives to the perturbation analysis schemes that update system parameters only at regeneration epochs (Ho and Cao [1991]). In Bhatnagar [2005], four adaptive Newton-based algorithms that estimate the Hessian in addition to the gradient of average cost have been developed in the simulation optimization setting. All of these use three timescales or step-size schedules and use four, three, two, and one simulation(s) of the system, respectively.

In this article, we develop three SF-based algorithms for simulation optimization. The first algorithm estimates only the gradient by using two-sided estimates. This is a direct extension of the algorithm in Bhatnagar and Borkar [2003] that is based on one-sided gradient estimates (similar to Katkovnik and Kulchitsky [1972]). Our next two algorithms are of the Newton type and estimate both the gradient and Hessian of average cost. We obtain two different estimates of the Hessian of the objective function that require one and two system simulations, respectively, and are obtained using convolutions of these with the multivariate normal density function. We show via an integration by parts argument (applied twice) that each of the above convolutions requires only simulated objective function values corresponding to certain perturbed parameters. The Hessian estimates are thus obtained in an elegant manner directly from the objective function simulations via a transformation that involves generating N independent $N(0, 1)$-distributed random variates at each update step (where N corresponds to the parameter dimension). One of our algorithms uses only one simulation at each update epoch and estimates both the gradient and Hessian of the objective function while the other does so with

two simulations and uses two-sided estimates. Note that the original method discussed in Katkovnik and Kulchitsky [1972], Rubinstein [1981], Styblinski and Opalski [1986], Chin [1997], and Styblinski and Tang [1990] only estimates the gradient. To the best of our knowledge, this is the first work that develops SF estimates for the Hessian and combines these together with previously known gradient estimates to develop two Newton-based SF algorithms for simulation optimization.

In the setting of simulation optimization that we consider, the objective function corresponds to the long-run average cost. Our gradient-based SF scheme requires two-timescale updates while our Newton-based algorithms use three timescales or step-size schedules in their recursions. Here, on the fastest timescale, data for the Hessian updates is aggregated and averaged. Next, the estimate of the Hessian is projected onto the space of positive definite and symmetric matrices. The inverse of the projected Hessian is then computed. Data for the gradient estimate is averaged on a slower timescale. Finally, on the slowest timescale recursion, the parameter is updated using estimates of the inverse of the projected Hessian matrix and the gradient of average cost. We provide a detailed convergence analysis for our algorithms. In particular, we show the unbiasedness of the Hessian estimates in the limit as the "spread parameter" goes to zero. We present numerical results using our algorithms over a setting involving a network of two $M/G/1$ queues with feedback and show performance comparisons with other algorithms developed in Bhatnagar [2005], Bhatnagar and Borkar [2003], and Bhatnagar et al. [2001, 2003], respectively. The setting for our experiments that we consider is similar to that in Bhatnagar [2005]. As emphasized before, our main contribution in this work is in the development of efficient Newton-based SF algorithms for simulation optimization that estimate the Hessian in addition to the gradient by requiring one and two system simulations, respectively. Our algorithm N-SF2 performs better in comparison with other algorithms in the literature over the setting considered here. We show unbiasedness of our Hessian estimates in the limit as the spread parameter goes to zero and prove convergence of all our algorithms.

The rest of the article is organized as follows: Section 2 presents the framework and some key assumptions. All our algorithms are presented in Section 3. We also present here a derivation of the gradient and Hessian estimators using the SF approach. Numerical experiments are presented in Section 4. The conclusions are given in Section 5. Finally, a detailed convergence analysis for our algorithms is presented in an Appendix at the end of the article.

2. FRAMEWORK

Let $\{X_n, n \geq 1\}$ be an $\mathcal{R}^d$-valued (for some given $d \geq 1$) parameterized Markov process with a tunable parameter θ that takes values in a given compact and convex set $C \subset \mathcal{R}^N$. We assume that, for any given $\theta \in C$, the process $\{X_n\}$ is ergodic Markov. Let $p(\theta, x, dy)$ and ν_θ, respectively, denote the transition kernel and stationary distribution of $\{X_n\}$ when θ is the operative parameter. Note that in discrete event systems (DES), state changes are effected upon occurrences of events at certain time points. The embedded state-valued process

of most (general) DES is Markov. Thus, Markov processes provide a powerful framework for the analysis of many DES; see, for instance, Cassandras [1993].

Let $h : \mathcal{R}^d \rightarrow \mathcal{R}^+$ be a given Lipschitz continuous cost function. Our aim is to find a $\theta \in C$ that minimizes the long-run average cost

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

The above limit exists because of ergodicity of the process $\{X_n\}$ for any given $\theta \in C$. We assume that $J(\theta)$ satisfies the following:

Assumption (A1). $J(\theta)$ is twice continuously differentiable in θ.

Note that (A1) is a standard requirement in most gradient search-based algorithms; see for instance, Bhatnagar [2005] and Spall [2000]. However, for our algorithms to work, we do not require (A1) as such. Our smoothed functional algorithms estimate not the gradient ∇J itself, but rather the convolution D_β of ∇J with a $N(0, \beta^2)$ probability density function. Even in the absence of (A1), we show that, for fixed $\beta > 0$, each of our algorithms converges to a point at which $D_\beta(\theta) = 0$. We use (A1) to show that both smoothed gradients and smoothed Hessians converge to their unsmoothed counterparts as $\beta \rightarrow 0$: $\| D_\beta(\theta) - \nabla J(\theta) \| \rightarrow 0$ and $\| D^2_\beta(\theta) - \nabla^2 J(\theta) \| \rightarrow 0$. Thus θ converges to a true local minimizer as $\beta \rightarrow 0$.

Suppose $\{\theta(n)\}$ is some sequence of random parameters obtained using (say) an iterative scheme on which the process $\{X_n\}$ depends. Let $\mathcal{H}_n = \sigma(\theta(m), X_m, m \leq n)$, $n \geq 1$ denote a sequence of associated σ-fields. We call $\{\theta(n)\}$ *nonanticipative* if for all Borel sets $A \subset \mathcal{R}^d$,

$$P(X_{n+1} \in A \mid \mathcal{H}_n) = p(\theta(n), X_n, A).$$

It can be easily seen that sequences $\{\theta(n)\}$ obtained using all our algorithms in the next section are nonanticipative. We shall assume the existence of a stochastic Liapunov function (below).

Assumption (A2). There exist $\epsilon_0 > 0$, $K \subset \mathcal{R}^d$ compact and $V \in C(\mathcal{R}^d)$ such that $\lim_{\|x\| \rightarrow \infty} V(x) = \infty$ and, under any nonanticipative $\{\theta(n)\}$,

(1) $\sup_n E[V(X_n)^2] < \infty$ and
(2) $E[V(X_{n+1}) \mid \mathcal{H}_n] \leq V(X_n) - \epsilon_0$, whenever $X_n \notin K$, $n \geq 0$.

Here and in the rest of the article, $\| \cdot \|$ shall denote the Euclidean norm. Also, for any matrix $A \in \mathcal{R}^{N \times N}$, its norm is defined as the induced matrix norm, also denoted using $\| \cdot \|$ and defined according to $\| A \| = \max_{\{x \in \mathcal{R}^N \mid \|x\| = 1\}} \| Ax \|$.

We require (A2) in order to ensure that the system remains stable under a tunable parameter. Assumption (A2) is used in the proofs of Lemmas A.1 and A.3, and is a standard requirement that has also been made in Bhatnagar and Borkar [1998] and Bhatnagar et al. [2001]. However, (A2) will not be required if the cost function $h(\cdot)$ is bounded in addition, as was the case studied in Bhatnagar [2005]. Suppose $\mathcal{P}(\mathcal{R}^d)$ is the space of all probability measures on $\mathcal{R}^d$ equipped with the Prohorov topology (which we briefly explain below;

see Chapter 2 of Borkar [1995] for details). A set $\{f_n, n \geq 1\}$ of bounded and continuous functions is called a *separating class* for $\mathcal{P}(\mathcal{R}^d)$ if for any probability measures $\mu, \nu \in \mathcal{P}(\mathcal{R}^d)$, $\int f_n d\mu = \int f_n d\nu$, $\forall n \geq 1$, implies $\mu = \nu$. Given a separating class of functions $\{f_n, n \geq 1\}$ of $\mathcal{P}(\mathcal{R}^d)$, we define a metric ρ on $\mathcal{P}(\mathcal{R}^d)$ as

$$\rho(\mu, \nu) = \sum_{n=1}^{\infty} |\int f_n d\mu - \int f_n d\nu|$$

for any $\mu, \nu \in \mathcal{P}(\mathcal{R}^d)$. The topology in $\mathcal{P}(\mathcal{R}^d)$ generated by ρ is called the *Prohorov topology*. The following result follows as a direct consequence of (A2) (see Lemma 2.1 of Bhatnagar and Borkar [1998] for a proof).

LEMMA 2.1. *$\{\nu_\theta, \theta \in C\}$ is compact in $\mathcal{P}(\mathcal{R}^d)$ and the map $\theta \rightarrow \nu_\theta : C \rightarrow \mathcal{P}(\mathcal{R}^d)$ is continuous.*

In Newton-based algorithms where the aim is to find a local minimum, one projects the Hessian estimate after each iteration onto the space of positive definite and symmetric matrices. This is required for the algorithm to progress along the negative gradient direction. In a small neighborhood of any local minimum, the Hessian matrix is expected to be positive definite. However, it need not be so in other portions of the search space. We let $P : \mathcal{R}^{N \times N} \rightarrow$ {positive definite and symmetric matrices} denote the projection operator that projects any $N \times N$–matrix to the space of positive definite and symmetric matrices. We assume $P(A) = A$, if A is positive definite and symmetric. In general, P can be characterized using various methods, for instance, via the *modified Choleski factorization procedure* (see Bertsekas [1999]), or the ones presented in Spall [2000] and Zhu and Spall [2002], respectively. For a matrix A, let $\{P(A)\}^{-1}$ denote the inverse of the matrix $P(A)$. We assume that the operator P satisfies the following:

Assumption (A3). If $\{A_n\}$ and $\{B_n\}$ are sequences of matrices in $\mathcal{R}^{N \times N}$ such that $\lim_{n \rightarrow \infty} \| A_n - B_n \| = 0$, then $\lim_{n \rightarrow \infty} \| P(A_n) - P(B_n) \| = 0$ as well. Further, for any sequence $\{C_n\}$ of matrices in $\mathcal{R}^{N \times N}$, if $\sup_n \| C_n \| < \infty$, then $\sup_n \| P(C_n) \|$, $\sup_n \| \{P(C_n)\}^{-1} \| < \infty$ as well.

Assumption (A3) is a technical requirement for the convergence analysis and has also been used in Bhatnagar [2005]. As argued in Bhatnagar [2005], the continuity requirement in (A3) can be easily imposed in the *modified Choleski factorization procedure* (see Bertsekas [1999], and the operators in Spall [2000]). Also the procedure in Zhu and Spall [2002] has been shown (there) to satisfy this requirement. For the other requirements in (A3), a sufficient condition is that, for some scalars $c_1, c_2 > 0$, the operator P be such that

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

Then all eigenvalues of $P(C_n)$, $\forall n$, lie between c_1 and c_2 and $\sup_n \| P(C_n) \|$, $\sup_n \| \{P(C_n)\}^{-1} \| < \infty$ (cf. Propositions A.9 and A.15 of Bertsekas [1999]). Most projection operators are seen to satisfy (2) (see Bhatnagar [2005] for a detailed discussion).

3. THE ALGORITHMS

In Bhatnagar and Borkar [2003], a one-sided SF algorithm that estimates only the gradient was presented. This was based on the Katkovnik-Kulchitsky [1972] scheme. We first begin by explaining the key idea there. For some scalar constant $\beta > 0$, let

$$D_{\beta,1}J(\theta) = \int G_\beta(\theta - \eta)\nabla_\eta J(\eta)d\eta \tag{3}$$

represent the convolution of the gradient of average cost ($J(\cdot)$) with the N-dimensional multivariate normal p.d.f.

$$G_\beta(\theta - \eta) = \frac{1}{(2\pi)^{N/2}\beta^N} \exp\left(-\frac{1}{2}\sum_{i=1}^{N}\frac{(\theta_i - \eta_i)^2}{\beta^2}\right),$$

where $\theta, \eta \in \mathcal{R}^N$ with $\theta \stackrel{\triangle}{=} (\theta_1, \ldots, \theta_N)^T$ and $\eta \stackrel{\triangle}{=} (\eta_1, \ldots, \eta_N)^T$. Integrating by parts in (3), it is easy to see that

$$D_{\beta,1}J(\theta) = \int \nabla_\theta G_\beta(\theta - \eta)J(\eta)d\eta = \int \nabla_\eta G_\beta(\eta)J(\theta - \eta)d\eta. \tag{4}$$

One can easily check that $\nabla_\eta G_\beta(\eta) = \frac{-\eta}{\beta^2}G_\beta(\eta)$. Substituting the last and $\eta' = \frac{\eta}{\beta}$ in (4), one obtains

$$D_{\beta,1}J(\theta) = \frac{1}{\beta}\int -\eta' \frac{1}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{N}(\eta_i')^2\right) J(\theta - \beta\eta')d\eta'. \tag{5}$$

In the above, we use the fact that $\eta = \beta\eta' = (\beta\eta_1', \ldots, \beta\eta_N')^T$ (written componentwise), and hence $d\eta = \beta^N d\eta_1' \cdots d\eta_N' = \beta^N d\eta'$. Upon substituting $\bar{\eta} = -\eta'$, one obtains

$$D_{\beta,1}J(\theta) = E\left[\frac{1}{\beta}\bar{\eta}J(\theta + \beta\bar{\eta})\right], \tag{6}$$

where the expectation above is taken with respect to the N-dimensional multivariate normal p.d.f.

$$G(\bar{\eta}) = \frac{1}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{N}(\bar{\eta}_i)^2\right).$$

The form of gradient estimator suggested by (6) (for M large and β small) is

$$\nabla J(\theta(n)) \approx \frac{1}{\beta}\frac{1}{M}\sum_{n=1}^{M}\eta(n)J(\theta(n) + \beta\eta(n)). \tag{7}$$

Here $\eta(n) \stackrel{\triangle}{=} (\eta_1(n), \ldots, \eta_N(n))^T$, with $\eta_i(n)$, $i = 1, \ldots, N$, $n \geq 0$, being independent $N(0, 1)$-distributed random variables. Before we proceed, we first describe the SF algorithm proposed in Bhatnagar and Borkar [2003]. We call it the *G-SF1 algorithm* to indicate that it is gradient based and requires one system-simulation. Subsequently, we develop and present the G-SF2, N-SF1, and N-SF2 algorithms, respectively. The G-SF2 algorithm is also gradient based and

uses a two-sided difference estimate for the gradient and hence requires two simulations. The N-SF1 and N-SF2 algorithms are Newton based and estimate both the gradient and Hessian using one and two simulations, respectively. While N-SF1 uses one-sided estimates for both gradient and Hessian, N-SF2 uses two-sided ones for both of these quantities.

Let $\{a(n)\}$ and $\{b(n)\}$ be two step-size sequences that satisfy the requirements

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

For $x = (x_1, \ldots, x_N)^T \in \mathcal{R}^N$, let $\Gamma(x) = (\Gamma_1(x_1), \ldots, \Gamma_N(x_N))^T$ represent the projection of x onto the set C. Let $L > 0$ be a given integer. It is generally observed (cf. Bhatnagar and Borkar [2003], Bhatnagar et al. [2001, 2003]; Bhatnagar [2005]) that the performance of the algorithm improves considerably when the parameter vector is updated once after a given number L of instants when $L > 1$. Let $Z_i(n)$, $i = 1, \ldots, N$, $n \geq 0$, be quantities defined via the recursions below that are used to estimate the average cost gradient. We now give the algorithm of Bhatnagar and Borkar [2003] we have labeled G-SFI.

The G-SF1 Algorithm

Step 0 (*Initialize*). Set $Z_1(0) = Z_2(0) = \cdots = Z_N(0) = 0$. Fix $\theta_1(0), \ldots, \theta_N(0)$ and let $\theta(0) \stackrel{\triangle}{=} (\theta_1(0), \ldots, \theta_N(0))^T$ denote the initial parameter vector. Fix L, M and β. Set $n := 0$ and $m := 0$, respectively. Generate i.i.d., $N(0, 1)$-distributed random variables $\eta_1(0), \eta_2(0), \ldots, \eta_N(0)$, and set $\eta(0) \stackrel{\triangle}{=} (\eta_1(0), \ldots, \eta_N(0))^T$.

Step 1. Generate the simulation X_{nL+m} governed with parameter $(\theta(n) + \beta\eta(n))$. Update for all $i = 1, \ldots, N$,

$$Z_i(nL + m + 1) = Z_i(nL + m) + b(n)\left(\frac{\eta_i(n)}{\beta} h(X_{nL+m}) - Z_i(nL + m)\right).$$

If $m = L - 1$, set $n := n + 1$, $m := 0$ and go to Step 2;
else, set $m := m + 1$ and repeat Step 1.

Step 2. For $i = 1, \ldots, N$, update $\theta_i(n)$ according to

$$\theta_i(n + 1) = \Gamma_i(\theta_i(n) - a(n)Z_i(nL)).$$

Set $n := n + 1$ and $\theta(n) \stackrel{\triangle}{=} (\theta_1(n), \ldots, \theta_N(n))^T$. If $n = M$, go to Step 3; else, generate i.i.d., $N(0, 1)$ distributed random variables $\eta_1(n), \ldots, \eta_N(n)$, independent of previous samples. Set $\eta(n) \stackrel{\triangle}{=} (\eta_1(n), \ldots, \eta_N(n))^T$ and go to Step 1.

Step 3 (*termination*). Terminate algorithm and output $\theta(M) \stackrel{\triangle}{=} (\theta_1(M), \ldots, \theta_N(M))^T$ as the final parameter vector.

A two-sided finite-difference form of SF estimates has been studied in Styblinski and Tang [1990] and Chin [1997]. The two-sided SF estimates are given by

$$D_{\beta,2}J(\theta) = E\left[\frac{\bar{\eta}}{2\beta}(J(\theta + \beta\bar{\eta}) - J(\theta - \beta\bar{\eta}))\right],$$

where, as with $D_{\beta,1}J(\theta)$, $\bar{\eta}$ is an N-vector of independent $N(0, 1)$ random variates and the above expectation is taken with respect to the distribution of $\bar{\eta}$.

The form of the gradient estimator is thus

$$\nabla J(\theta(n)) \approx \frac{1}{2\beta}\frac{1}{M}\sum_{n=1}^{M}\eta(n)(J(\theta(n)+\beta\eta(n))-J(\theta(n)-\beta\eta(n))), \tag{8}$$

for M large and β small. Here $\eta(n) \stackrel{\triangle}{=} (\eta_1(n),\ldots,\eta_N(n))^T$ is a vector of independent $N(0,1)$-distributed random variables as before. In the simulation optimization setting, this suggests the following form for the corresponding gradient-based algorithm that we refer to as *G-SF2*.

The G-SF2 Algorithm

Step 0 (*Initialize*). Set $Z_1(0) = Z_2(0) = \cdots = Z_N(0) = 0$. Fix $\theta_1(0),\ldots,\theta_N(0)$ and let $\theta(0) \stackrel{\triangle}{=} (\theta_1(0),\ldots,\theta_N(0))^T$ denote the initial parameter vector. Fix L, M and β. Set $n := 0$ and $m := 0$, respectively. Generate i.i.d., $N(0,1)$-distributed random variables $\eta_1(0),\ldots,\eta_N(0)$, and set $\eta(0) \stackrel{\triangle}{=} (\eta_1(0),\ldots,\eta_N(0))^T$.

Step 1. Generate two parallel simulations X^1_{nL+m} and X^2_{nL+m} that are governed with parameters $(\theta(n)+\beta\eta(n))$ and $(\theta(n)-\beta\eta(n))$, respectively. Update for all $i = 1,\ldots,N$,

$$Z_i(nL+m+1) = Z_i(nL+m) + b(n)\left(\frac{\eta_i(n)}{2\beta}(h(X^1_{nL+m})-h(X^2_{nL+m})) - Z_i(nL+m)\right).$$

If $m = L-1$, set $n := n+1, m := 0$ and go to Step 2; else, set $m := m+1$ and repeat Step 1.

Step 2. For $i = 1,\ldots,N$, update $\theta_i(n)$ according to

$$\theta_i(n+1) = \Gamma_i(\theta_i(n) - a(n)Z_i(nL)).$$

Set $n := n+1$ and $\theta(n) \stackrel{\triangle}{=} (\theta_1(n),\ldots,\theta_N(n))^T$. If $n = M$, go to Step 3; else, generate i.i.d., $N(0,1)$ distributed random variables $\eta_1(n),\ldots,\eta_N(n)$, independent of previous samples. Set $\eta(n) \stackrel{\triangle}{=} (\eta_1(n),\ldots,\eta_N(n))^T$ and go to Step 1.

Step 3 (*Termination*). Terminate algorithm and output $\theta(M) \stackrel{\triangle}{=} (\theta_1(M),\ldots,\theta_N(M))^T$ as the final parameter vector.

In what follows, we shall first extend the above ideas to get a one-sided estimate $D^2_{\beta,1}J(\theta)$ on the Hessian $\nabla^2 J(\theta)$ and later provide a two-sided Hessian estimate as well. We also develop the corresponding Newton-based SF algorithms N-SF1 and N-SF2, respectively. Let

$$D^2_{\beta,1}J(\theta) = \int G_\beta(\theta-\eta)\nabla^2_\eta J(\eta)d\eta, \tag{9}$$

with $G_\beta(\theta-\eta)$ defined as before. Upon integrating by parts, one obtains

$$D^2_{\beta,1}J(\theta) = \int \nabla_\theta G_\beta(\theta-\eta)\nabla_\eta J(\eta)d\eta.$$

Now

$$\nabla_\theta G_\beta(\theta-\eta) = -\frac{(\theta-\eta)}{\beta^2}G_\beta(\theta-\eta).$$

Hence,

$$D^2_{\beta,1}J(\theta) = -\frac{1}{\beta^2}\int (\theta-\eta)G_\beta(\theta-\eta)\nabla_\eta J(\eta)d\eta = -\frac{1}{\beta^2}\int \nabla_\theta((\theta-\eta)G_\beta(\theta-\eta))J(\eta)d\eta.$$

The last equality above is obtained via another operation involving integration by parts. By a change of variables, one obtains

$$D^2_{\beta,1}J(\theta) = -\frac{1}{\beta^2}\int \nabla_\eta(\eta G_\beta(\eta))J(\theta-\eta)d\eta. \tag{10}$$

Before we proceed further, we first evaluate $\nabla_\eta(\eta G_\beta(\eta)) = \nabla_\eta((\eta_1 G_\beta(\eta), \ldots, \eta_N G_\beta(\eta))$. Note that $\nabla_\eta(\eta G_\beta(\eta))$ equals

$$\begin{bmatrix} \nabla_{\eta_1}(\eta_1 G_\beta(\eta)) & \nabla_{\eta_2}(\eta_1 G_\beta(\eta)) & \cdots & \nabla_{\eta_N}(\eta_1 G_\beta(\eta)) \\ \nabla_{\eta_1}(\eta_2 G_\beta(\eta)) & \nabla_{\eta_2}(\eta_2 G_\beta(\eta)) & \cdots & \nabla_{\eta_N}(\eta_2 G_\beta(\eta)) \\ \cdots & \cdots & \cdots & \cdots \\ \nabla_{\eta_1}(\eta_N G_\beta(\eta)) & \nabla_{\eta_2}(\eta_N G_\beta(\eta)) & \cdots & \nabla_{\eta_N}(\eta_N G_\beta(\eta)) \end{bmatrix}$$

$$= \begin{bmatrix} \left(1-\frac{\eta_1^2}{\beta^2}\right) & -\frac{\eta_1\eta_2}{\beta^2} & \cdots & -\frac{\eta_1\eta_N}{\beta^2} \\ -\frac{\eta_2\eta_1}{\beta^2} & \left(1-\frac{\eta_2^2}{\beta^2}\right) & \cdots & -\frac{\eta_2\eta_N}{\beta^2} \\ \cdots & \cdots & \cdots & \cdots \\ -\frac{\eta_N\eta_1}{\beta^2} & -\frac{\eta_N\eta_2}{\beta^2} & \cdots & \left(1-\frac{\eta_N^2}{\beta^2}\right) \end{bmatrix} G_\beta(\eta).$$

Let $\hat{H}(\eta)$ denote the matrix above that multiplies $G_\beta(\eta)$. Then from (10), we have

$$D^2_{\beta,1}J(\theta) = -\frac{1}{\beta^2}\int \hat{H}(\eta)G_\beta(\eta)J(\theta-\eta)d\eta.$$

As with the calculation of the gradient, let $\eta' = \eta/\beta$. Then $d\eta = \beta^N d\eta'$. Hence (10) becomes

$$D^2_{\beta,1}J(\theta) = \frac{1}{\beta^2}\int \bar{H}(\eta')\left(\frac{1}{(2\pi)^{N/2}}\exp\left(-\frac{1}{2}\sum_{i=1}^N(\eta_i')^2\right)\right)J(\theta-\beta\eta')d\eta'. \tag{11}$$

In the above,

$$\bar{H}(\eta') = \begin{bmatrix} ((\eta_1')^2-1) & \eta_1'\eta_2' & \cdots & \eta_1'\eta_N' \\ \eta_2'\eta_1' & ((\eta_2')^2-1) & \cdots & \eta_2'\eta_N' \\ \cdots & \cdots & \cdots & \cdots \\ \eta_N'\eta_1' & \eta_N'\eta_2' & \cdots & ((\eta_N')^2-1) \end{bmatrix}. \tag{12}$$

Note that η_i', $i = 1, \ldots, N$ are independent $N(0,1)$ distributed random variables. Now since η_i' and $-\eta_i'$ have the same distribution, one obtains

$$D^2_{\beta,1}J(\theta) = E\left[\frac{1}{\beta^2}\bar{H}(\bar{\eta})J(\theta+\beta\bar{\eta})\right],$$

where the expectation above is taken with respect to the N-dimensional multivariate normal p.d.f. $G(\bar{\eta})$ (as before) corresponds to the random vector of N

independent $N(0,1)$-distributed random variables. Hence the form of the estimator for $\nabla^2 J(\theta(n))$ suggested by the above is (for M large and β small)

$$\nabla^2 J(\theta(n)) \approx \frac{1}{\beta^2}\frac{1}{M}\sum_{n=1}^{M} \bar{H}(\eta(n))J(\theta(n)+\beta\eta(n)). \tag{13}$$

Here $\eta(n) = (\eta_1(n), \ldots, \eta_N(n))^T$ is a vector of independent $N(0,1)$-distributed random variables. We now propose a three-timescale Newton-based algorithm to implement the above. Let $\{a(n)\}$, $\{b(n)\}$ and $\{c(n)\}$ be three step-size sequences. We assume that these satisfy the following requirements.

Assumption (A4).

$$\sum_n a(n) = \sum_n b(n) = \sum_n c(n) = \infty, \quad \sum_n (a(n)^2 + b(n)^2 + c(n)^2) < \infty, \tag{14}$$

$$a(n) = o(c(n)) \text{ and } c(n) = o(b(n)). \tag{15}$$

Note that (14) are standard requirements on step-size sequences in stochastic approximation algorithms. From (15), $a(n) \to 0$, the fastest, and $b(n) \to 0$, the slowest, among the three step-size schedules. Thus, in the algorithm below, the timescale corresponding to $\{b(n)\}$ is the fastest while that corresponding to $\{a(n)\}$ is the slowest since, beyond some finite integer N_0, increments in recursions governed by $\{b(n)\}$ are uniformly the largest while those governed by $\{a(n)\}$ are uniformly the smalle amongs the three types of recursions. Hence, one expects that recursions corresponding to $\{b(n)\}$ would asymptotically track the stable equilibrium points of the associated ODEs, the fastest, albeit with a possibly higher variance in their iterates. The timescale corresponding to $\{c(n)\}$ is faster than the one corresponding to $\{a(n)\}$ but slower than that corresponding to $\{b(n)\}$. In both our algorithms N-SF1 and N-SF2, the Hessian is updated on the fastest timescale (using the step-size sequence $\{b(n)\}$) while the parameter is updated on the slowest scale (with step-size sequence $\{a(n)\}$). This is because the Hessian recursion needs to converge the fastest as the inverse of the (converged) Hessian update is required. The gradient is updated using the step-size sequence $\{c(n)\}$ as it should ideally move on a scale in between the Hessian and parameter updates. Further, both the Hessian and gradient updates should converge before a parameter update step. This helps in keeping the parameter iterates stable. We now present the N-SF1 algorithm.

The N-SF1 Algorithm

Step 0 (*Initialize*). Set $Z_k(0) = Z_{i,j}(0) = 0, \forall k, i, j \in \{1, \ldots, N\}$. Fix $\theta_k(0), k = 1, \ldots, N$ and let $\theta(0) \stackrel{\triangle}{=} (\theta_1(0), \ldots, \theta_N(0))^T$ denote the initial parameter vector. Fix L, M and β. Set $n := 0$ and $m := 0$, respectively. Generate i.i.d., $N(0,1)$-distributed random variables $\eta_1(0), \eta_2(0), \ldots, \eta_N(0)$, and set $\eta(0) \stackrel{\triangle}{=} (\eta_1(0), \ldots, \eta_N(0))^T$.

Step 1. Generate the simulation X_{nL+m} governed with parameter $(\theta(n) + \beta\eta(n))$. For $i, j, k = 1, \ldots, N$, $j < k$, update

$$Z_{i,i}(nL+m+1) = Z_{i,i}(nL+m) + b(n)\left(\frac{\eta_i^2(n)-1}{\beta^2}h(X_{nL+m}) - Z_{i,i}(nL+m)\right),$$

$$Z_{j,k}(nL+m+1) = Z_{j,k}(nL+m) + b(n)\left(\frac{\eta_j(n)\eta_k(n)}{\beta^2}h(X_{nL+m}) - Z_{j,k}(nL+m)\right).$$

For $j > k$, set $Z_{j,k}(nL+m+1) = Z_{k,j}(nL+m+1)$. Next for $l = 1, \ldots, N$, update

$$Z_l(nL+m+1) = Z_l(nL+m) + c(n)\left(\frac{\eta_l(n)}{\beta}h(X_{nL+m}) - Z_l(nL+m)\right).$$

If $m = L-1$, set $n := n+1$ and $m := 0$. Next form the matrix $H(nL) = P([[Z_{j,k}(nL)]]_{j,k=1}^N)$ and compute its inverse $M(nL) = [[M_{j,k}(nL)]]_{j,k=1}^N \stackrel{\triangle}{=} H(nL)^{-1}$, and go to Step 2; else, set $m := m+1$ and repeat Step 1.

Step 2. For $i = 1, \ldots, N$, update $\theta_i(n)$ according to

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - a(n)\sum_{k=1}^{N} M_{i,k}(nL)Z_k(nL)\right).$$

Set $n := n+1$ and $\theta(n) \stackrel{\triangle}{=} (\theta_1(n), \ldots, \theta_N(n))^T$. If $n = M$, go to Step 3; else, generate i.i.d., $N(0,1)$-distributed random variables $\eta_1(n), \ldots, \eta_N(n)$, independent of previous samples. Set $\eta(n) \stackrel{\triangle}{=} (\eta_1(n), \ldots, \eta_N(n))^T$ and go to Step 1.

Step 3 (*Termination*). Terminate algorithm and output $\theta(M) \stackrel{\triangle}{=} (\theta_1(M), \ldots, \theta_N(M))^T$ as the final parameter vector.

In Step 1, since the Hessian is symmetric and is derived from the matrix $\bar{H}(\eta')$ in (12), we only update $Z_{i,j}(nL+m)$ for $i < j$ in Step 1 and set $Z_{i,j}(nL+m+1) = Z_{j,i}(nL+m+1)$ for $i > j$. Again note that Algorithm N-SF1 requires only one simulation.

Next, we propose a two-sided estimate for the Hessian that uses two parallel simulations. This will be used together with a finite difference estimate of the gradient (similar to G-SF2) in our next algorithm N-SF2. The form of the smoothed functional ($D^2_{\beta,2}J(\theta)$) for the Hessian that we propose is the following:

$$D^2_{\beta,2}J(\theta) = E\left[\frac{1}{2\beta^2}\bar{H}(\bar{\eta})(J(\theta+\beta\bar{\eta}) + J(\theta-\beta\bar{\eta}))\right].$$

The form of the proposed two-sided estimator of the Hessian is thus

$$\nabla^2 J(\theta(n)) \approx \frac{1}{2\beta^2}\frac{1}{M}\sum_{n=1}^{M}\bar{H}(\eta(n))(J(\theta(n)+\beta\eta(n)) + J(\theta(n)-\beta\eta(n))), \qquad (16)$$

where $\eta(n) \stackrel{\triangle}{=} (\eta_1(n), \ldots, \eta_N(n))^T$ is a vector of $N(0,1)$ random variates. Algorithm N-SF2 uses the same gradient estimate as G-SF2.

The N-SF2 Algorithm

Step 0 (*Initialize*). Set $Z_k(0) = Z_{i,j}(0) = 0, \forall k, i, j \in \{1, \ldots, N\}$. Fix $\theta_k(0), k = 1, \ldots, N$ and let $\theta(0) \stackrel{\triangle}{=} (\theta_1(0), \ldots, \theta_N(0))^T$ denote the initial parameter vector. Fix L, M and β. Set $n := 0$ and $m := 0$, respectively. Generate i.i.d., $N(0,1)$-distributed random variables $\eta_1(0), \eta_2(0), \ldots, \eta_N(0)$, and set $\eta(0) \stackrel{\triangle}{=} (\eta_1(0), \ldots, \eta_N(0))^T$.

Step 1. Generate two parallel simulations X^1_{nL+m} and X^2_{nL+m} that are governed with parameters $(\theta(n)+\beta\eta(n))$ and $(\theta(n)-\beta\eta(n))$, respectively. For $i, j, k = 1, \ldots, N$, $j < k$, update

$$\begin{aligned} Z_{i,i}(nL+m+1) \;=\; & Z_{i,i}(nL+m) + b(n)\left(\frac{\eta_i^2(n)-1}{2\beta^2}\left(h(X^1_{nL+m}) + h(X^2_{nL+m})\right)\right. \\ & \left. - Z_{i,i}(nL+m)\right), \end{aligned}$$

$$Z_{j,k}(nL+m+1) = Z_{j,k}(nL+m) + b(n)\left(\frac{\eta_j(n)\eta_k(n)}{2\beta^2}\left(h\left(X^1_{nL+m}\right) + h\left(X^2_{nL+m}\right)\right) - Z_{j,k}(nL+m)\right).$$

For $j > k$, set $Z_{j,k}(nL+m+1) = Z_{k,j}(nL+m+1)$. Next, for $l = 1, \ldots, N$, update

$$Z_l(nL+m+1) = Z_l(nL+m) + c(n)\left(\frac{\eta_l(n)}{2\beta}\left(h\left(X^1_{nL+m}\right) - h\left(X^2_{nL+m}\right)\right) - Z_l(nL+m)\right).$$

If $m = L-1$, set $n := n+1$ and $m := 0$. Next form the matrix $H(nL) = P([[Z_{j,k}(nL)]]^N_{j,k=1})$ and compute its inverse $M(nL) = [[M_{j,k}(nL)]]^N_{j,k=1} \stackrel{\triangle}{=} H(nL)^{-1}$, and go to Step 2; else, set $m := m+1$ and repeat Step 1.

Step 2. For $i = 1, \ldots, N$, update $\theta_i(n)$ according to

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - a(n)\sum_{k=1}^{N} M_{i,k}(nL)Z_k(nL)\right).$$

Set $n := n+1$ and $\theta(n) \stackrel{\triangle}{=} (\theta_1(n), \ldots, \theta_N(n))^T$. If $n = M$, go to Step 3; else, generate i.i.d., $N(0,1)$-distributed random variables $\eta_1(n), \ldots, \eta_N(n)$, independent of previous samples. Set $\eta(n) \stackrel{\triangle}{=} (\eta_1(n), \ldots, \eta_N(n))^T$ and go to Step 1.

Step 3 (Termination). Terminate algorithm and output $\theta(M) \stackrel{\triangle}{=} (\theta_1(M), \ldots, \theta_N(M))^T$ as the final parameter vector.

3.1 The Jacobi Variants of N-SF1 and N-SF2

For purposes of implementation in our numerical experiments, we consider variants of both the N-SF1 and N-SF2 algorithms wherein the entire Hessian is not updated at each epoch but only its diagonal elements are. The cross-diagonal elements in the Hessian are simply set to zero. Thus projecting the resulting (by abuse of terminology) Hessian update onto the set of positive definite and symmetric matrices is straightforward as all one needs to do is to project each element of the diagonal matrix to the positive half-line. Computing and projecting the entire Hessian at each update epoch onto the set of positive definite and symmetric matrices, and computing its inverse takes a large computational effort. Moreover, the above procedure has also been proposed in Spall [2000] for the case of high-dimensional parameters. A similar procedure has been adopted in Bhatnagar [2005] as well for the algorithms therein. The resulting algorithms that we implement are thus the Jacobi variants of the proposed algorithms. We also describe, in the Appendix, the convergence analysis for the Jacobi variants of algorithms N-SF1 and N-SF2 in addition to that of the original algorithms.

For our numerical experiments, we project each diagonal element of the (resulting) Hessian update to the interval $[0.1, \infty)$. The requirements in Assumption (A3) can be seen to hold in this case since the projection operator is continuous. Further, (2) holds in this setting since all eigenvalues are greater than or equal to 0.1. Also, one can show, as in Lemma A.3, that the iterates of the above scaling matrix and hence of the projected matrix are uniformly upper bounded with probability 1.

4. NUMERICAL EXPERIMENTS

We show performance comparisons of our algorithms G-SF2 and the Jacobi variants of N-SF1 and N-SF2 (which we again refer to as N-SF1 and N-SF2, respectively, for simplicity) with the G-SF1 algorithm of Bhatnagar and Borkar [2003], the one- and two-simulation Gradient SPSA algorithms (G-SPSA1 and G-SPSA2, respectively) of Bhatnagar et al. [2003], and the one- and two-simulation Newton based SPSA algorithms (N-SPSA1 and N-SPSA2, respectively) of Bhatnagar [2005]. For the sake of completeness, we first briefly describe the above Gradient and Newton SPSA algorithms.

4.1 Gradient and Newton SPSA Algorithms

We begin by describing the one- and two-simulation Gradient SPSA algorithms, G-SPSA1 and G-SPSA2, respectively.

4.1.1 *Gradient SPSA (G-SPSA) Algorithms.* These algorithms estimate only the gradient of the objective. Let $\triangle(n) \stackrel{\triangle}{=} (\triangle_1(n), \ldots, \triangle_N(n))^T$ be a vector of i.i.d random variables $\triangle_1(n), \ldots, \triangle_N(n)$, $n \geq 0$, each of which is Bernoulli distributed with $\triangle_i(n) = \pm 1$ w.p. 1/2, $i = 1, \ldots, N$, $n \geq 0$. More general distributions for the above (see Spall [1992]) may, however, be chosen. Let $\delta > 0$ be a given small constant. Let $\{X_l^+\}$ ($\{X_l^+\}$ and $\{X_l^-\}$) denote the simulation(s) that is (are) governed by parameter sequence(s) $\{\theta_l^+\}$ ($\{\theta_l^+\}$ and $\{\theta_l^-\}$, respectively). Here $\theta_l^+ = \theta(n) + \delta\triangle(n)$ and $\theta_l^- = \theta(n) - \delta\triangle(n)$ for $n = [\frac{l}{L}]$, with $L \geq 1$ a given fixed integer as before. Let $\tilde{Z}(0) = 0$ in the recursions below.

4.1.1.1 *G-SPSA1.*

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_i(n)}\right)\right), \tag{17}$$

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

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

4.1.1.2 *G-SPSA2.*

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - a(n)\left(\frac{\tilde{Z}(nL)}{2\delta\triangle_i(n)}\right)\right), \tag{19}$$

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

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

4.1.2 *Newton SPSA (N-SPSA) Algorithms.* These algorithms estimate both the gradient and Hessian of the objective. Let $\delta_1, \delta_2 > 0$ be given small constants. Let $\triangle(n) \stackrel{\triangle}{=} (\triangle_1(n), \ldots, \triangle_N(n))^T$ and $\hat{\triangle}(n) \stackrel{\triangle}{=} (\hat{\triangle}_1(n), \ldots, \hat{\triangle}_N(n))^T$ be independent sequences of i.i.d random variables $\triangle_1(n), \ldots, \triangle_N(n)$, and $\hat{\triangle}_1(n)$, $\ldots, \hat{\triangle}_N(n)$, respectively, with $\triangle_i(n), \hat{\triangle}_i(n) = \pm 1$ w.p. 1/2, $i = 1, \ldots, N$, $n \geq 0$. Suppose $\{X^{++}(l)\}$ ($\{X^{++}(l)\}$ and $\{X^+(l)\}$) denote the simulation(s) that is (are) governed by parameter sequence(s) $\{\theta_l^{++}\}$ ($\{\theta_l^{++}\}$ and $\{\theta_l^+\}$, respectively). Here, $\theta_l^+ = \theta(n) + \delta_1\triangle(n)$ and $\theta_l^{++} = \theta(n) + \delta_1\triangle(n) + \delta_2\hat{\triangle}(n)$ for $n = [\frac{l}{L}]$ as before. Let $\tilde{Z}(0) = 0$.

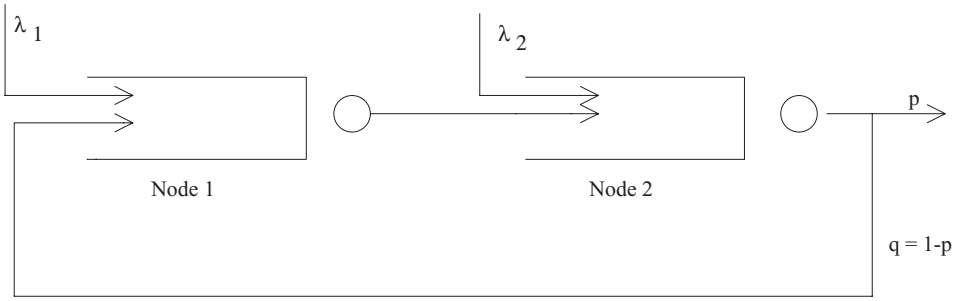

Fig. 1. Queueing network.

4.1.2.1 *N-SPSA1.* For $n \geq 0$, $m = 0, 1, \ldots, L-1$,

$$\tilde{Z}(nL+m+1) = \tilde{Z}(nL+m) + b(n)(h(X^{++}(nL+m)) - \tilde{Z}(nL+m)). \tag{21}$$

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

$$H_{j,i}(n+1) = H_{j,i}(n) + c(n)\left(\frac{\tilde{Z}(nL)}{\delta_1\delta_2\triangle_i(n)\hat{\triangle}_j(n)} - H_{j,i}(n)\right). \tag{22}$$

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

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) - a(n)\sum_{k=1}^{N} M_{i,k}(n)\left(\frac{\tilde{Z}(nL)}{\delta_2\hat{\triangle}_k(n)}\right)\right). \tag{23}$$

4.1.2.2 *N-SPSA2.* The only difference here with N-SPSA1 is that (21) is replaced with

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

Recursions (22)–(23) are now the same as before.

As with the N-SF1 and N-SF2 algorithms, we consider for the purposes of implementation the Jacobi variants of N-SPSA1 and N-SPSA2. Next we present our numerical setting and results.

4.2 Numerical Setting and Results

We consider a two-node network of $M/G/1$ queues with feedback as shown in Figure 1. The setting here is the same as that considered in Bhatnagar [2005]. A somewhat similar setting is also considered in Bhatnagar et al. [2001] and Bhatnagar et al. [2003].

We allow nodes 1 and 2 to be fed with independent Poisson arrival processes with rates $\lambda_1 = 0.2$ and $\lambda_2 = 0.1$, respectively. Customers after service at Node 1 enter Node 2. Upon completion of service at Node 2, a customer either leaves the system with probability $p = 0.4$ or joins Node 1 with probability $q = 0.6$, independently of other customers. We assume the cost function to be the sum of waiting times of individual customers at the two nodes. Let $A = [[A_{j,j'}]]$ be

a given positive definite and symmetric matrix. Let the vector of parameters to be tuned be denoted $\theta = (\theta^1, \theta^2)$, where $\theta^i = (\theta^i_1, \ldots, \theta^i_M)^T$, $i = 1, 2$. The service time processes $\{S^i_n(\theta^i)\}$ at the two nodes $i = 1, 2$, depend on the above parameter in the manner $S^i_n(\theta^i) = U^i_n(1 + \sum_{j=1}^{M} \sum_{j'=1}^{M} (\theta^i_j(n) - \bar{\theta}^i_j)(\theta^i_{j'}(n) - \bar{\theta}^i_{j'})A_{j,j'})/R_i$, $i = 1, 2, n \geq 1$. Here U^1_n, U^2_n are independent samples from the uniform distribution $U(0, 1)$. Also, we select $R_1 = 10$ and $R_2 = 20$, respectively. Thus, in vector-matrix notation, $S^i_n(\theta^i) = U^i_n(1+ (\theta^i - \bar{\theta}^i)^T A(\theta^i - \bar{\theta}^i))$. Since A is positive definite and symmetric, $S^i_n(\theta^i)$ have the lowest values for $\theta^i = \bar{\theta}^i$. This choice of θ^i minimizes the cost. Thus, $\bar{\theta}^i_1, \ldots, \bar{\theta}^i_M$, $i = 1, 2$, represent the target parameter components to which each algorithm must converge. Note that if we have $-\log(U^i_n)$ in place of U^i_n above, then $S^i_n(\theta^i)$ would correspond, via the inverse transform method for generating random variates, to a sample from the exponential distribution with parameter $R_i/(1 + \sum_{j=1}^{M} \sum_{j'=1}^{M} (\theta^i_j(n) - \bar{\theta}^i_j)(\theta^i_{j'}(n) - \bar{\theta}^i_{j'})A_{j,j'})$.

We consider two different choices for M: $M = 2$ and 25, respectively. The dimensions of the corresponding parameters are thus $N = 4$ and 50. We consider the Euclidean distance $d(\theta(n), \bar{\theta}) \stackrel{\triangle}{=} (\sum_{i=1}^{2} \sum_{j=1}^{M} (\theta^i_j(n) - \bar{\theta}^i_j)^2)^{1/2}$ of $\theta(n)$, $n \geq 1$ from $\bar{\theta}$ as our measure of performance. For $M = 25$, we let $A = I$ (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. For the cost to be minimized, one expects $d(\theta(n), \bar{\theta}) \to 0$ as $n \to \infty$. We show performance comparisons of the various algorithms using the above metric.

In all the experiments whose results are shown in Tables I–VII, we ran all the algorithms for a total of 12×10^5 simulations. Thus, when $L = 100$ (as with the experiments whose results are shown in Tables I, II, III, VI, and VII, respectively), for each of the one (two)-simulation algorithms, a total of 12,000 (6000) parameter updates are obtained. On a Pentium 5 PC with Linux operating system, each of the four SPSA algorithms take about 20 (5) s for one simulation run for the case $N = 50$ ($N = 4$), while each of the four SF algorithms take about 1 min (20 s). The SF algorithms take slightly longer than their SPSA counterparts since generating Gaussian random variates is more computationally expensive than generating Bernoulli variates. All simulations are run independently with 20 different initial seeds.

For our first experiment, we set $L = 100$, $\beta = 0.1$, $\delta = 0.1$, and $\delta_1 = \delta_2 = 0.1$ as the given sensitivity parameters in the various algorithms. In Figures 2 and 3, we plot the trajectories for the mean values of $d(\theta(n), \bar{\theta})$ obtained from the 20 independent simulation runs, using all the algorithms for both $N = 4$ and 50, respectively. The mean and standard deviation values obtained from these simulations upon termination in all the algorithms are then presented in Table I. We select step-sizes $\{a(n)\}$, $\{b(n)\}$, and $\{c(n)\}$ according to $a(n) = \frac{1}{n}$, $b(n) = \frac{1}{n^{2/3}}$, and $c(n) = \frac{1}{n^{3/4}}$, respectively, $n \geq 1$, with $a(0) = b(0) = c(0) = 1$, in the various algorithms. Note, however, that the gradient-based algorithms G-SF1, G-SF2, G-SPSA1, and G-SPSA2 do not use step sizes $c(n)$. From these experiments, we observe that, when $N = 4$, G-SPSA2 gives the best results and is closely followed by N-SF2. However, for $N = 50$, N-SF2 shows the best results and is in fact significantly better than the other algorithms. Also, for this case ($N = 50$), G-SPSA2 and N-SPSA2 are slightly better as compared

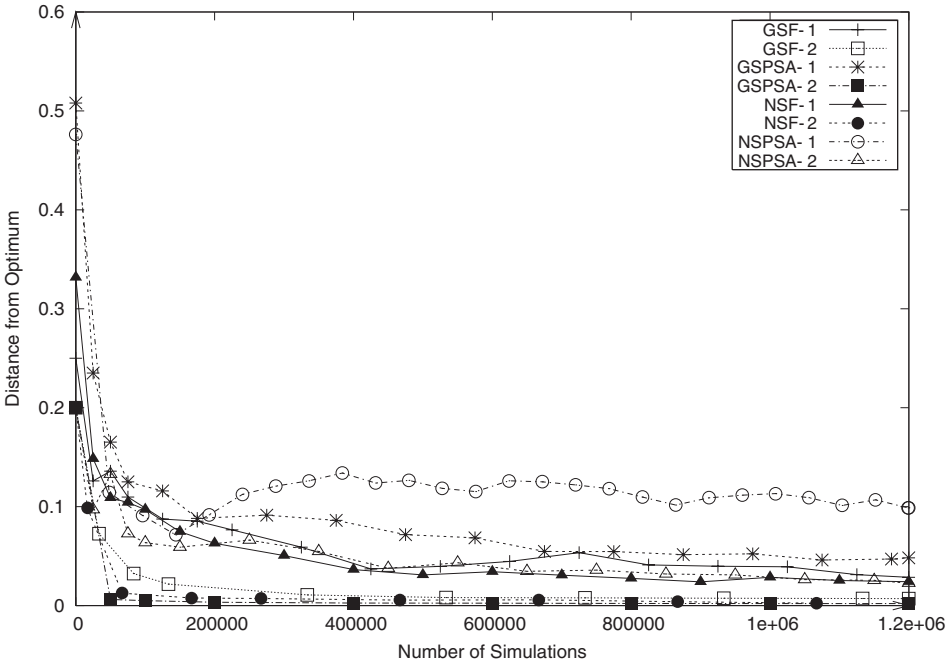

Fig. 2. Convergence behavior of the algorithms for $N = 4$.

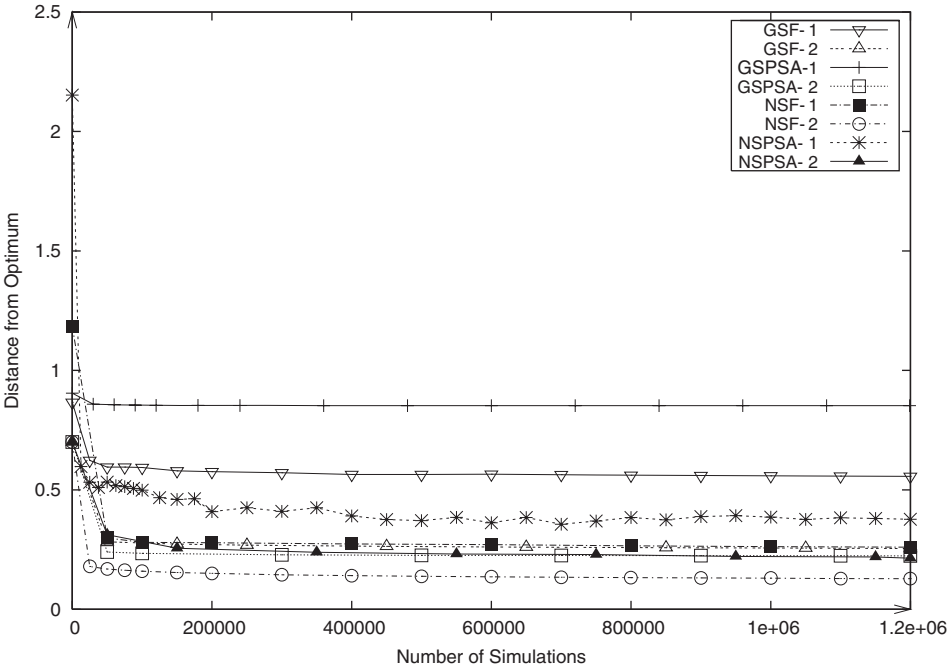

Fig. 3. Convergence behavior of the algorithms for $N = 50$.

Table I. Performance after 12×10^5 Simulations

Algorithm	$d(\theta(n), \bar{\theta})$ $N = 4$	$d(\theta(n), \bar{\theta})$ $N = 50$
N-SF2	0.0030 ± 0.0009	0.1278 ± 0.0088
G-SF2	0.0070 ± 0.0010	0.2546 ± 0.0168
G-SPSA2	0.0020 ± 0.0004	0.2237 ± 0.0312
N-SPSA2	0.0227 ± 0.0096	0.2139 ± 0.0283
N-SF1	0.0242 ± 0.0112	0.2598 ± 0.0189
G-SF1	0.0285 ± 0.0132	0.5567 ± 0.0692
G-SPSA1	0.0483 ± 0.0184	0.8525 ± 0.0466
N-SPSA1	0.1072 ± 0.0220	0.3768 ± 0.0228

Table II. Performance of SF Algorithms for Different Values of β When $N = 50$

β	$d(\theta(n), \bar{\theta})$ for G-SF2	$d(\theta(n), \bar{\theta})$ for N-SF2	$d(\theta(n), \bar{\theta})$ for G-SF1 $N = 50$	$d(\theta(n), \bar{\theta})$ for N-SF1
0.001	0.3363 ± 0.0219	0.1584 ± 0.0142	0.5990 ± 0.0812	0.3310 ± 0.0643
0.01	0.3384 ± 0.0228	0.1381 ± 0.0109	0.5208 ± 0.0616	0.3084 ± 0.0372
0.05	0.2723 ± 0.0189	0.1187 ± 0.0124	0.5165 ± 0.0412	0.2594 ± 0.0409
0.10	0.2546 ± 0.0168	0.1278 ± 0.0088	0.5567 ± 0.0692	0.2598 ± 0.0189
0.15	0.2314 ± 0.0112	0.1430 ± 0.0064	0.5497 ± 0.0273	0.2765 ± 0.0220
0.20	0.2670 ± 0.0205	0.1381 ± 0.0102	0.5889 ± 0.0366	0.2408 ± 0.0411
0.30	0.2532 ± 0.0211	0.1292 ± 0.0171	0.6344 ± 0.0298	0.3036 ± 0.0213
0.40	0.2891 ± 0.0168	0.1533 ± 0.0287	0.6886 ± 0.0401	0.3103 ± 0.0108
0.50	0.2718 ± 0.0305	0.1596 ± 0.0191	0.6728 ± 0.0349	0.3111 ± 0.0235
0.75	0.2865 ± 0.0336	0.1734 ± 0.0335	0.6911 ± 0.0382	0.3189 ± 0.0188
1.00	0.2991 ± 0.0378	0.1739 ± 0.0299	0.6739 ± 0.0316	0.3237 ± 0.0316

to G-SF2 and N-SF1. Among the one-simulation algorithms, N-SF1 shows the best results in both cases. G-SPSA1 and G-SF1 do not show good performance when $N = 50$.

Next, we study comparisons in performance of various algorithms under different settings of the sensitivity parameters. In Tables II–III, we compare performance of the SF algorithms for varying values of the *spread parameter* β, for $N = 50$ and $N = 4$, respectively, with the rest of the setting parameters as before. Here it can be seen that N-SF2 consistently outperforms the other algorithms for $N = 50$. The same is also true for $N = 4$, except for a couple of cases: $\beta = 0.01$ and $\beta = 0.05$, in which G-SF2 is better. Note also that G-SF2 performs better than N-SF1 in seven out of the 11 cases for $N = 50$ but in only three out of the 11 cases for $N = 4$. G-SF1 does not show good performance in comparison with the other algorithms when $N = 50$. For $N = 4$, however, G-SF1 is better than N-SF1 for $\beta = 0.05$ and $\beta = 0.15$, respectively. It is observed in most algorithms that performance deteriorates when β is either too small or is increased beyond a point. This is due to the fact that low values of β lead to high variability and overall bias in the estimates (cf. Styblinski and Tang [1990]) while high values result in inaccuracies in the gradient and Hessian estimates. As discussed in Section A.1 of the Appendix, the performance of N-SF2, however, does not degrade much when β is made low or high.

In Tables IV–V, performance comparisons of all algorithms are shown for $N = 50$ for different values of the *averaging parameter* L. The value of β is set at 0.1 for all SF algorithms. Here also, N-SF2 can be seen to perform better in

Table III. Performance of SF Algorithms for Different Values of When $N = 4$

β β	$d(\theta(n), \bar{\theta})$ for G-SF2	$d(\theta(n), \bar{\theta})$ for N-SF2	$d(\theta(n), \bar{\theta})$ for G-SF1	$d(\theta(n), \bar{\theta})$ for N-SF1
0.001	0.1903 ± 0.0466	0.1389 ± 0.0327	0.3105 ± 0.0761	0.1457 ± 0.0514
0.01	0.0163 ± 0.0140	0.0219 ± 0.0202	0.2062 ± 0.0522	0.1066 ± 0.0278
0.05	0.0033 ± 0.0067	0.0091 ± 0.0038	0.0172 ± 0.009	0.0765 ± 0.0109
0.10	0.0070 ± 0.0010	0.0030 ± 0.0009	0.0285 ± 0.0132	0.0242 ± 0.0112
0.15	0.0413 ± 0.0134	0.0094 ± 0.0071	0.0283 ± 0.0196	0.0308 ± 0.0141
0.20	0.0978 ± 0.0360	0.0284 ± 0.0128	0.0824 ± 0.0208	0.0327 ± 0.0106
0.30	0.1038 ± 0.0344	0.0713 ± 0.0269	0.1237 ± 0.0185	0.0512 ± 0.0276
0.40	0.1307 ± 0.0319	0.0886 ± 0.0301	0.1620 ± 0.0368	0.1090 ± 0.0395
0.50	0.1473 ± 0.0298	0.1217 ± 0.0392	0.1861 ± 0.0504	0.1139 ± 0.0251
0.75	0.1524 ± 0.0277	0.1231 ± 0.0210	0.1929 ± 0.0356	0.1235 ± 0.0299
1.00	0.1728 ± 0.0300	0.1229 ± 0.0253	0.2001 ± 0.0327	0.1348 ± 0.0388

Table IV. Performance of SF Algorithms for Different Values of L When $N = 50$

L	$d(\theta(n), \bar{\theta})$ for G-SF2	$d(\theta(n), \bar{\theta})$ for N-SF2	$d(\theta(n), \bar{\theta})$ for G-SF1	$d(\theta(n), \bar{\theta})$ for N-SF1
1	0.3119 ± 0.0211	0.2679 ± 0.0178	0.5973 ± 0.0277	0.3040 ± 0.0169
10	0.2568 ± 0.0303	0.0495 ± 0.0227	0.5264 ± 0.0198	0.2267 ± 0.0124
50	0.2494 ± 0.0189	0.0999 ± 0.0143	0.4656 ± 0.0262	0.1956 ± 0.0197
100	0.2546 ± 0.0168	0.1278 ± 0.0088	0.5567 ± 0.0692	0.2598 ± 0.0189
200	0.2642 ± 0.0277	0.1093 ± 0.0093	0.5586 ± 0.0348	0.2604 ± 0.0201
500	0.2718 ± 0.0395	0.1286 ± 0.0161	0.5806 ± 0.0365	0.2750 ± 0.0188
1000	0.2789 ± 0.0264	0.1481 ± 0.0200	0.6009 ± 0.0259	0.3112 ± 0.0223

comparison with all the algorithms and in some cases by more than an order of magnitude; see, for instance the cases of $L = 10$ and $L = 50$. Among algorithms other than N-SF2, N-SF1 shows the best results overall here. The performance of G-SF2 and G-SPSA2 is almost similar, with G-SF2 marginally better in most cases here. The performance of all four algorithms is not good when $L = 1$ is used. Performance improves as L is increased, implying that an additional averaging over the two-timescale averaging is desirable. Similar observations have also been made in Bhatnagar et al. [2001, 2003] and Bhatnagar [2005] as well. Note also that performance deteriorates somewhat when L is increased beyond a point implying that excessive averaging is also not desirable for overall system performance.

Next, we study performance comparisons between the N-SF and N-SPSA algorithms for the case of $N = 50$, for varying *step-size parameters*, and also study the impact of step-size on performance. We set $L = 100$ for all algorithms. All other setting parameters are the same as before. In Table VI, we use step-sizes $a(n) = 1/n$, $b(n) = 1/n^{0.55}$, and $c(n) = 1/n^{\alpha}$, respectively, where α is varied between the values of 0.55 and 1.0. One may use other choices for $b(n)$. For instance, $b(n)$ here can be chosen as $b(n) = 1/n^{\gamma}$ for any γ satisfying $0.5 < \gamma < 1$. It can be seen that performance deteriorates when α is very close to 0.55 or 1.0, namely, when step-size $c(n)$ is close to $b(n)$ or $a(n)$, respectively. In Table VII, we use $a(n) = 1/n$, $c(n) = 1/n^{0.75}$, and $b(n) = 1/n^{\gamma}$, respectively, where γ is varied between 0.55 and 0.75. It can be seen that, for all the algorithms, performance is good for lower values of γ and deteriorates as γ is brought close

Table V. Performance of SPSA Algorithms for Different Values of L When $N = 50$

L	$d(\theta(n), \bar{\theta})$ for G-SPSA2	$d(\theta(n), \bar{\theta})$ for N-SPSA2	$d(\theta(n), \bar{\theta})$ for G-SPSA1	$d(\theta(n), \bar{\theta})$ for N-SPSA1
1	0.3165 ± 0.0254	0.3304 ± 0.0309	1.1531 ± 0.0192	0.4122 ± 0.0346
10	0.2627 ± 0.0218	0.3286 ± 0.0221	0.7100 ± 0.0451	0.3466 ± 0.0225
50	0.2557 ± 0.0193	0.3027 ± 0.0235	0.8090 ± 0.0384	0.3689 ± 0.0156
100	0.2237 ± 0.0312	0.2139 ± 0.0283	0.8525 ± 0.0466	0.3768 ± 0.0228
200	0.2706 ± 0.0287	0.2339 ± 0.0128	0.7575 ± 0.0340	0.3084 ± 0.0215
500	0.2781 ± 0.0199	0.2236 ± 0.0300	0.8238 ± 0.0278	0.3997 ± 0.0329
1000	0.2738 ± 0.0249	0.3107 ± 0.0227	0.8671 ± 0.0333	0.4018 ± 0.0281

Table VI. Performance of Newton-Based Algorithms for $c(n) = 1/n^{\alpha}$, $a(n) = 1/n$ and $b(n) = 1/n^{0.55}$ When $N = 50$

α	$d(\theta(n), \bar{\theta})$ for N-SF2	$d(\theta(n), \bar{\theta})$ for N-SF1	$d(\theta(n), \bar{\theta})$ for N-SPSA2	$d(\theta(n), \bar{\theta})$ for N-SPSA1
0.55	0.1812 ± 0.0220	0.3115 ± 0.0378	0.2903 ± 0.0334	0.4212 ± 0.0409
0.60	0.1439 ± 0.0098	0.2774 ± 0.0291	0.2712 ± 0.0329	0.4066 ± 0.0373
0.65	0.0921 ± 0.0083	0.2362 ± 0.0295	0.2569 ± 0.0241	0.3947 ± 0.0236
0.70	0.1007 ± 0.0186	0.2467 ± 0.0259	0.2298 ± 0.0188	0.3975 ± 0.0381
0.75	0.1123 ± 0.0104	0.1905 ± 0.0178	0.2381 ± 0.0232	0.3544 ± 0.0206
0.80	0.0992 ± 0.0217	0.2292 ± 0.0182	0.2496 ± 0.0287	0.3603 ± 0.0212
0.85	0.1281 ± 0.0248	0.2661 ± 0.0289	0.2970 ± 0.0313	0.3530 ± 0.0308
0.90	0.1397 ± 0.0274	0.2868 ± 0.0334	0.2561 ± 0.0272	0.3441 ± 0.0253
0.95	0.1415 ± 0.0309	0.2738 ± 0.0272	0.2695 ± 0.0200	0.3856 ± 0.0298
1.00	0.1597 ± 0.0315	0.3259 ± 0.0526	0.2817 ± 0.0296	0.3979 ± 0.0440

to 0.75, thereby again suggesting the need for a clear separation in timescales. In both Tables VI and VII, N-SF2 shows much better performance than the other algorithms. Also, N-SPSA2 shows better results compared with N-SF1 in most cases here (six out of the 10 cases in Table VI and five out of seven in Table VII).

Finally, for the same setting as in Table I, for $N = 50$, we measured the time required by each algorithm for $d(\theta(n), \bar{\theta})$ to become less than 0.1. It took about 3 min for N-SF2, while for N-SF1 and G-SF2 it took close to 14 min each. N-SPSA2 and G-SPSA2 took about 9 and 10 min, respectively, while N-SPSA1 took about 32 min. G-SPSA1 and G-SF1, on the other hand, did not reach the target even after 5 h.

5. CONCLUSIONS

We developed three SF-based stochastic approximation algorithms for simulation optimization. While one of our algorithms estimates the gradient of the objective by using two-sided gradient estimates, the other two estimate both the gradient and Hessian of the objective function by using one and two simulations, respectively. The original SF algorithm due to Katkovnik and Kulchitsky [1972] estimates only the gradient using one-sided estimates. A two-sided gradient estimate was proposed and used in Styblinski and Tang [1990] and Chin [1997]. Using similar methods as in Katkovnik and Kulchitsky [1972] and Styblinski and Opalski [1986], we derive two SF-based Hessian estimates that use (the same) one and two simulations (as for gradient estimates). These are

Table VII. Performance of Newton-Based Algorithms for $b(n) = 1/n^{\gamma}$, $a(n) = 1/n$ and $c(n) = 1/n^{0.75}$ When $N = 50$

γ	$d(\theta(n), \bar{\theta})$ for N-SF2	$d(\theta(n), \bar{\theta})$ for N-SF1	$d(\theta(n), \bar{\theta})$ for N-SPSA2	$d(\theta(n), \bar{\theta})$ for N-SPSA1
0.55	0.1123 ± 0.0104	0.1905 ± 0.0178	0.2381 ± 0.0232	0.3544 ± 0.0206
0.58	0.0971 ± 0.0124	0.2517 ± 0.0252	0.2412 ± 0.0277	0.3563 ± 0.0148
0.64	0.0822 ± 0.0183	0.2357 ± 0.0161	0.2496 ± 0.0318	0.3671 ± 0.0408
0.66	0.1278 ± 0.0088	0.2156 ± 0.0197	0.2139 ± 0.0283	0.3689 ± 0.0156
0.70	0.1349 ± 0.0175	0.2677 ± 0.0275	0.2288 ± 0.0209	0.3897 ± 0.0299
0.72	0.1497 ± 0.0199	0.2796 ± 0.0202	0.2554 ± 0.0306	0.3898 ± 0.0393
0.75	0.1672 ± 0.0247	0.3102 ± 0.0429	0.2781 ± 0.0369	0.4110 ± 0.0518

then used in our Newton-based algorithms. We showed numerical results over a setting involving a network of two $M/G/1$ queues with feedback and compared the performance of our algorithms with other one- and two-simulation gradient and Newton-based algorithms in the literature. We studied several experiments by varying the setting parameters and step-sizes. Here N-SF2 is seen to consistently show the best performance among all algorithms. Also, G-SF2 and N-SF1 performed favorably in many cases over other algorithms in the literature. In particular, N-SF1 showed the best results among one-simulation algorithms and did comparably well in many cases over other well-known two-simulation algorithms. A detailed convergence analysis of our algorithms is shown in the Appendix.

We use a constant value for the *spread parameter* β in our algorithm. This results in convergence to a near-optimum point. As future work, one could develop algorithms where β goes to zero over a timescale that is the slowest among all timescales in order to ensure convergence to an optimum. In Bhatnagar and Borkar [2003], the use of a rapidly mixing chaotic random number generator for generating Gaussian random variates was seen to improve performance in a related gradient-based SF algorithm. Similar random number generators may also be used for our algorithms. Finally, algorithms that use perturbation variates other than Gaussian, for instance, Cauchy and uniform, as suggested in Styblinski and Tang [1990], may be worth exploring in the context of SF algorithms.

APPENDIX: CONVERGENCE ANALYSIS

We first show the detailed convergence analysis of algorithm N-SF1. Next, the changes in the above analysis required for N-SF2 are presented. The convergence of G-SF2 is then briefly indicated. Finally, we briefly present the analysis for the Jacobi variants of N-SF1 and N-SF2.

A.1 Convergence Analysis of N-SF1

Let $\mathcal{F}(l) = \sigma(\tilde{\theta}_i(p), X_p, \tilde{\eta}_i(p), p \leq l, i = 1, \ldots, N), l \geq 1$, denote σ-fields generated by the quantities above. Here $\tilde{\theta}_i(p) = \theta_i(n)$ and $\tilde{\eta}_i(p) = \eta_i(n)$, respectively, for $i = 1, \ldots, N$, $nL \leq p \leq (n+1)L - 1$. Define $\{\tilde{b}(n)\}$ and $\{\tilde{c}(n)\}$ as follows: for $n \geq 0$, $\tilde{b}(n) = b([\frac{n}{L}])$ and $\tilde{c}(n) = c([\frac{n}{L}])$, where $[\frac{n}{L}]$ denotes the integer part of $\frac{n}{L}$. Note that $\{\tilde{b}(n)\}$ ($\{\tilde{c}(n)\}$) corresponds to the natural timescale over which the

Hessian (data) averaging step should be analyzed. It is easy to see that

$$\sum_n \tilde{b}(n) = \sum_n \tilde{c}(n) = \infty, \quad \sum_n (\tilde{b}(n)^2 + \tilde{c}(n)^2) < \infty, \tag{25}$$

$$\tilde{c}(n) = o(\tilde{b}(n)), \quad a(n) = o(\tilde{c}(n)). \tag{26}$$

In fact $\{b(n)\}$ ($\{c(n)\}$) goes to zero faster than $\{\tilde{b}(n)\}$ ($\{\tilde{c}(n)\}$) does. Thus, $\{\tilde{b}(n)\}$ ($\{\tilde{c}(n)\}$) corresponds to an even faster step-size sequence compared with $\{b(n)\}$ ($\{c(n)\}$). The timescale difference of $\{a(n)\}$ with $\{\tilde{b}(n)\}$ ($\{\tilde{c}(n)\}$) is thus higher compared to the same with $\{b(n)\}$ ($\{c(n)\}$). This is seen to improve performance. The Hessian and gradient update recursions in Step 1 of the algorithm can be rewritten as follows: for $i, j, k = 1, \ldots, N$, $j < k$, update

$$Z_{i,i}(p+1) = Z_{i,i}(p) + \tilde{b}(p)\left(\frac{\tilde{\eta}_i^2(p) - 1}{\beta^2} h(X_p) - Z_{i,i}(p)\right), \tag{27}$$

$$Z_{j,k}(p+1) = Z_{j,k}(p) + \tilde{b}(p)\left(\frac{\tilde{\eta}_j(p)\tilde{\eta}_k(p)}{\beta^2} h(X_p) - Z_{j,k}(p)\right). \tag{28}$$

For $j > k$, set $Z_{j,k}(p+1) = Z_{k,j}(p+1)$. Further, for $l = 1, \ldots, N$,

$$Z_l(p+1) = Z_l(p) + \tilde{c}(p)\left(\frac{\tilde{\eta}_l(p)}{\beta} h(X_p) - Z_l(p)\right). \tag{29}$$

Define sequences $\{M_{l,l}(p)\}$, $\{M_{i,j}(p)\}$, $l, i, j \in \{1, \ldots, N\}$, $i \neq j$, as follows: for $l = 1, \ldots, N$,

$$M_{l,l}(p) = \sum_{m=1}^{p} \tilde{b}(m)\left(\frac{\tilde{\eta}_l^2(m) - 1}{\beta^2} h(X_m) - E\left[\frac{\tilde{\eta}_l^2(m) - 1}{\beta^2} h(X_m) \mid \mathcal{F}(m-1)\right]\right).$$

Further, for $i, j \in \{1, \ldots, N\}$, we have

$$M_{i,j}(p) = \sum_{m=1}^{p} \tilde{b}(m)\left(\frac{\tilde{\eta}_i(m)\tilde{\eta}_j(m)}{\beta^2} h(X_m) - E\left[\frac{\tilde{\eta}_i(m)\tilde{\eta}_j(m)}{\beta^2} h(X_m) \mid \mathcal{F}(m-1)\right]\right).$$

LEMMA A.1. *The sequences* $\{M_{l,l}(p), \mathcal{F}(p)\}$ *and* $\{M_{i,j}(p), \mathcal{F}(p)\}$, $l, i, j = 1, \ldots, N$, $i \neq j$, *are almost surely convergent martingale sequences.*

PROOF. We consider first the sequence $\{M_{l,l}(p), \mathcal{F}(p)\}$. It is easy to see that, almost surely, $E[M_{l,l}(p+1) \mid \mathcal{F}(p)] = M_{l,l}(p)$, for all $p \geq 0$. Now note that

$$\begin{aligned} E\big[M_{l,l}^2(p)\big] \leq \frac{C_p}{\beta^4} \sum_{m=1}^{p} \tilde{b}^2(m) \big(& E\big[\big(\tilde{\eta}_l^2(m) - 1\big)^2 h^2(X_m) \\ & + E^2\big[\big(\tilde{\eta}_l^2(m) - 1\big) h(X_m) \mid \mathcal{F}(m-1)\big]\big]\big) \end{aligned}$$

for some constant $C_p > 0$ (that however depends on p). For the second term on the right hand side above, note that, almost surely,

$$E^2\big[\big(\tilde{\eta}_l^2(m) - 1\big) h(X_m) \mid \mathcal{F}(m-1)\big] \leq E\big[\big(\tilde{\eta}_l^2(m) - 1\big)^2 h^2(X_m) \mid \mathcal{F}(m-1)\big],$$

by the conditional Jensen's inequality. Hence,

$$E\left[M_{l,l}^2(p)\right] \leq \frac{2C_p}{\beta^4}\sum_{m=1}^{p}\tilde{b}^2(m)E\left[\left(\tilde{\eta}_l^2(m)-1\right)^2 h^2(X_m)\right]$$
$$\leq \frac{2C_p}{\beta^4}\sum_{m=1}^{p}\tilde{b}^2(m)E\left[\left(\tilde{\eta}_l^2(m)-1\right)^2\right]^{1/2}E\left[h^4(X_m)\right]^{1/2}$$

by the Cauchy-Schwartz inequality. Since, $h(\cdot)$ is a Lipschitz continuous function, we have

$$|h(X_m)| - |h(0)| \leq |h(X_m) - h(0)| \leq K \parallel X_m \parallel,$$

where $K > 0$ is the Lipschitz constant. Thus,

$$|h(X_m)| \leq C_1(1+ \parallel X_m \parallel)$$

for $C_1 = \max(K, |h(0)|) < \infty$. Hence, one gets

$$E[h^4(X_m)] \leq C_2(1+ \parallel X_m \parallel^4)$$

for (constant) $C_2 = 8C_1^4$. As a consequence of (A2), $\sup_m \parallel X_m \parallel^4 < \infty$. Thus, $E[M_{l,l}^2(p)] < \infty$, for all $p \geq 1$. One can also see that $M_{l,l}(p)$ are integrable random variables. Now note that

$$\sum_p E[(M_{l,l}(p+1) - M_{l,l}(p))^2 \mid \mathcal{F}(p)]$$
$$\leq \sum_p \tilde{b}^2(p+1)\left(E\left[\left(\frac{\tilde{\eta}_l^2(p+1)-1}{\beta^2}h(X_{p+1})\right)^2 \mid \mathcal{F}(p)\right]\right.$$
$$\left. + E[E^2\left[\frac{\tilde{\eta}_l^2(p+1)-1}{\beta^2}h(X_{p+1}) \mid \mathcal{F}(p)\right] \mid \mathcal{F}(p)\right]\Bigg)$$
$$\leq \sum_p 2\tilde{b}^2(p+1)E\left[\left(\frac{\tilde{\eta}_l^2(p+1)-1}{\beta^2}h(X_{p+1})\right)^2 \mid \mathcal{F}(p)\right],$$

almost surely. The last inequality above again follows from the conditional Jensen's inequality. It can now be easily seen as before, using (A2), that

$$\sup_p \frac{1}{\beta^2}E\left[\left(\left(\tilde{\eta}_l^2(p+1)-1\right)h(X_{p+1})\right)^2 \mid \mathcal{F}(p)\right] < \infty \text{ w.p.1.}$$

Hence, using (A4), it can be seen that

$$\sum_p E[(M_{l,l}(p+1) - M_{l,l}(p))^2 \mid \mathcal{F}(p)] < \infty$$

almost surely. Thus, by the martingale convergence theorem, $\{M_{l,l}(p)\}$ are almost surely convergent martingale sequences. A similar proof settles the claim for $\{M_{i,j}(p)\}$ as well. □

Define $\{s(n)\}$, $\{t(n)\}$ and $\{r(n)\}$ as follows: $s(0) = t(0) = r(0) = 0$, $s(n) = \sum_{i=0}^{n-1} a(i)$, $t(n) = \sum_{i=0}^{n-1} \tilde{b}(i)$, and $r(n) = \sum_{i=0}^{n-1} \tilde{c}(i)$, $n \geq 1$, respectively. Then the timescale corresponding to $\{s(n)\}$ is the slowest timescale while the one corresponding to $\{t(n)\}$ is the fastest. Also, the timescale obtained from $\{r(n)\}$ falls in between the above two. Let $Y(nL) \stackrel{\triangle}{=} [[Z_{j,k}(nL)]]_{j,k=1}^{N}$ and $Z(nL) \stackrel{\triangle}{=} (Z_1(nL), \ldots, Z_N(nL))^T$. Thus, the matrix $H(nL)$ in the algorithm corresponds to $P(Y(nL))$.

Consider the following system of ordinary differential equations (ODEs):

$$\dot{\theta}(t) = 0, \tag{30}$$

$$\dot{Z}(t) = 0, \tag{31}$$

$$\dot{Y}(t) = D_{\beta,1}^2 J(\theta) - Y(t). \tag{32}$$

We now recall a key result from Hirsch [1989] stated here as Lemma A.2. Consider an ODE

$$\dot{x}(t) = F(x(t)), \tag{33}$$

which has an asymptotically stable attracting set G. Let G^ϵ denote the ϵ-neighborhood of G, namely, $G^\epsilon = \{x \mid \exists x' \in G \text{ such that } \| x - x' \| \leq \epsilon\}$. For $\tau > 0$, $\mu > 0$, we call $y(\cdot)$ a (τ, μ)-perturbation of (33) 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 (33) such that $\sup_{t \in [\tau_i, \tau_{i+1}]} |x^i(t) - y(t)| < \mu$. We have

LEMMA A.2. *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* (33) *converges to G^ϵ.*

Note that, in vector-matrix notation, the Hessian update recursions (27)–(28) can be written as

$$Y(p+1) = Y(p) + \tilde{b}(p)(\bar{H}(\tilde{\eta}(p))h(X_p) - Y(p)). \tag{34}$$

We have the following:

LEMMA A.3. *The sequence of Hessian updates $\{Y(p)\}$ is uniformly bounded with probability* 1.

PROOF. Note that (34) can be rewritten as

$$\begin{aligned} Y(p+1) = {} & Y(p) + \tilde{b}(p)(E[\bar{H}(\tilde{\eta}(p))h(X_p) \mid \mathcal{F}(p-1)] - Y(p)) \\ & + \tilde{b}(p)(\bar{H}(\tilde{\eta}(p))h(X_p) - E[\bar{H}(\tilde{\eta}(p))h(X_p) \mid \mathcal{F}(p-1)]). \end{aligned}$$

From Lemma A.1, we have that, almost surely,

$$\sum_p \tilde{b}(p)(\bar{H}(\tilde{\eta}(p))h(X_p) - E[\bar{H}(\tilde{\eta}(p))h(X_p) \mid \mathcal{F}(p-1)]) < \infty.$$

Hence it is sufficient to show the boundedness of the following recursion:

$$\bar{Y}(p+1) = \bar{Y}(p) + \tilde{b}(p)(E[\bar{H}(\tilde{\eta}(p))h(X_p) \mid \mathcal{F}(p-1)] - \bar{Y}(p)),$$

with $\bar{Y}(0) = Y(0)$. As in the proof of Lemma A.1, it can again be seen that $\sup_p E[\bar{H}(\tilde{\eta}(p))h(X_p) \mid \mathcal{F}(p-1)] < \infty$ with probability 1. Now since $\tilde{b}(p) \to 0$

as $p \to \infty$, there exists an integer p_0 such that, for all $p \geq p_0$, $0 \leq \tilde{b}(p) \leq 1$. Hence, for all $p \geq p_0$, $\bar{Y}(p+1)$ is a convex combination of $\bar{Y}(p)$ and a quantity that is almost surely uniformly bounded. The claim follows. □

Consider now functions $\hat{Y}(t)$ defined according to $\hat{Y}(t(n)) = Y(nL)$ with the maps $t \to \hat{Y}(t)$ 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 $Y^n(t)$, $t \in I_n$, $n \geq 0$, that are obtained as trajectories of the following ODEs:

$$\dot{Y}^n(t) = D^2_{\beta,1} J(\theta) - Y^n(t), \tag{35}$$

with $Y^n(T_n) = \hat{Y}(t(m_n)) = Y(m_n L)$. Note that, in (35), the parameter θ is held fixed, namely, $\theta(t) \equiv \theta$ since the parameter update recursion in Step 2 of the algorithm can be rewritten as

$$\theta_i(n+1) = \Gamma_i(\theta_i(n) + \tilde{b}(n)\xi_1(n)), \tag{36}$$

where $\xi_1(n) = o(1)$ as $a(n) = o(\tilde{b}(n))$. In other words, when viewed from the timescale of $\{\tilde{b}(n)\}$, the parameter update recursion is quasistatic. Now a standard argument using Gronwall's inequality can be used to show (see for instance Bhatnagar et al. [2001]) the following:

LEMMA A.4. $\lim_{n\to\infty} \sup_{t \in I_n} \| Y^n(t) - \hat{Y}(t) \| = 0$ *w.p.1.*

Next, we have the following:

LEMMA A.5. *Given $T, \gamma > 0$, $((\theta(t(n)+\cdot), Z(t(n)+\cdot), \hat{Y}(t(n)+\cdot))$ is a bounded (T, γ)-perturbation of (30)–(32) for n sufficiently large.*

PROOF. Observe that the gradient update recursions in Step 1 of the algorithm can be written as

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

where $\xi_2(n) = o(1)$ since $\tilde{c}(n) = o(\tilde{b}(n))$. Also, the parameter update recursion can be written as in (36). The rest now follows from Lemma A.4. □

COROLLARY A.6. $\| Y(nL) - D^2_{\beta,1} J(\theta(n)) \| \to 0$ *w.p. 1, as* $n \to \infty$.

PROOF. The claim follows by Lemma A.2 applied on (32) for every $\epsilon > 0$. □

The next result shows that the Hessian estimates are unbiased in the limit as $\beta \to 0$.

PROPOSITION A.7. $\| D^2_{\beta,1} J(\theta(n)) - \nabla^2 J(\theta(n)) \| \to 0$ *as* $\beta \to 0$.

PROOF. Recall that

$$D^2_{\beta,1} J(\theta(n)) = E\left[\frac{1}{\beta^2} \bar{H}(\eta(n)) J(\theta(n) + \beta\eta(n))\right],$$

where $\eta(n) = (\eta_1(n), \ldots, \eta_N(n))^T$ is a vector of independent $N(0,1)$ random variates and the expectation is taken with respect to the density of $\eta(n)$. For ease of exposition, we drop the dependence of $\theta(n)$ and $\eta(n)$ on the index n and simply denote these quantities as θ and η, respectively. Hence, using a Taylor series expansion of $J(\theta + \beta\eta)$ around θ, one obtains

$$D^2_{\beta,1}J(\theta) = E\left[\frac{1}{\beta^2}\bar{H}(\eta)(J(\theta)\right] + \beta\eta^T\nabla J(\theta) + \frac{\beta^2}{2}\eta^T\nabla^2 J(\theta)\eta + o(\beta^2))$$
$$= \frac{1}{\beta^2}E[\bar{H}(\eta)J(\theta)] + \frac{1}{\beta}E[\bar{H}(\eta)\eta^T\nabla J(\theta)] + \frac{1}{2}E[\bar{H}(\eta)\eta^T\nabla^2 J(\theta)\eta] + O(\beta). \quad (37)$$

Now observe that $E[\bar{H}(\eta)] = 0$ (the zero matrix). Here, $E[\bar{H}(\eta)]$ is a matrix of expectations of individual elements of $\bar{H}(\eta)$. Hence the first term on the right-hand side of (37) equals zero. Now consider the second term on the right-hand side of (37). Note that

$$E[\bar{H}(\eta)\eta^T\nabla J(\theta)] = E\begin{bmatrix} (\eta_1^2-1)\eta^T\nabla J(\theta) & \eta_1\eta_2\eta^T\nabla J(\theta) & \cdots & \eta_1\eta_N\eta^T\nabla J(\theta) \\ \eta_2\eta_1\eta^T\nabla J(\theta) & (\eta_2^2-1)\eta^T\nabla J(\theta) & \cdots & \eta_2\eta_N\eta^T\nabla J(\theta) \\ \cdots & \cdots & \cdots & \cdots \\ \eta_N\eta_1\eta^T\nabla J(\theta) & \eta_N\eta_2\eta^T\nabla J(\theta) & \cdots & (\eta_N^2-1)\eta^T\nabla J(\theta) \end{bmatrix}. \quad (38)$$

Consider the first term (corresponding to the first row and first column) above. Note that

$$E[(\eta_1^2-1)\eta^T\nabla J(\theta)] = E[(\eta_1^3-\eta_1, \eta_1^2\eta_2-\eta_2, \ldots, \eta_1^2\eta_N-\eta_N)^T\nabla J(\theta)] = 0.$$

Similarly all other terms in (38) can be seen to be equal to zero as well. We use here the facts that $E[\eta_1] = E[\eta_1^3] = 0$ and $E[\eta_1^2] = 1$. Also, η_i is independent of η_j for all $i \neq j$. Hence the second term on the right-hand side of (37) equals zero as well. Consider now the third term on the right-hand side of (37). Note that

$$\frac{1}{2}E[\bar{H}(\eta)\eta^T\nabla^2 J(\theta)\eta] =$$
$$\frac{1}{2}E\begin{bmatrix} (\eta_1^2-1)\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j & \eta_1\eta_2\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j & \cdots & \eta_1\eta_N\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j \\ \eta_2\eta_1\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j & (\eta_2^2-1)\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j & \cdots & \eta_2\eta_N\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j \\ \cdots & \cdots & \cdots & \cdots \\ \eta_N\eta_1\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j & \eta_N\eta_2\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j & \cdots & (\eta_N^2-1)\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j \end{bmatrix}. \quad (39)$$

Consider now the term corresponding to the first row and first column above. Note that

$$E[(\eta_1^2-1)\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j = E\left[\eta_1^2\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j\right] - E\left[\sum_{i,j=1}^N \nabla_{ij}J(\theta)\eta_i\eta_j\right]. \quad (40)$$

The first term on the right-hand side of (40) equals

$$E[\eta_1^4\nabla_{11}J(\theta)] + E\left[\sum_{i=j,i\neq 1}\eta_1^2\eta_i^2\nabla_{ij}J(\theta)\right] + E\left[\sum_{i\neq j,i\neq 1}\eta_1^2\eta_i\eta_j\nabla_{ij}J(\theta)\right]$$
$$= 3\nabla_{11}J(\theta) + \sum_{i=j,i\neq 1}\nabla_{ij}J(\theta),$$

since $E[\eta_1^4] = 3$. The second term on right-hand side of (40) equals $-\sum_{i=1}^{N} \nabla_{ii} J(\theta)$. Adding the above two terms, one obtains

$$E\left[(\eta_1^2 - 1)\sum_{i,j=1}^{N} \nabla_{ij} J(\theta)\eta_i\eta_j\right] = 2\nabla_{11} J(\theta).$$

Consider now the term in the first row and second column of the matrix in (39). Note that

$$E\left[\eta_1\eta_2 \sum_{i,j=1}^{N} \nabla_{ij} J(\theta)\eta_i\eta_j\right] = 2E\left[\eta_1^2\eta_2^2\nabla_{12} J(\theta)\right] + E\left[\sum_{(i,j)\notin\{(1,2),(2,1)\}} \eta_1\eta_2\eta_i\eta_j \nabla_{ij} J(\theta)\right]$$

$$= 2\nabla_{12} J(\theta).$$

Proceeding in a similar manner, it is easy to verify that the (i, j)th term $(i, j \in \{1, \ldots, N\})$ in the matrix in (39) equals $2\nabla_{ij} J(\theta)$. Substituting the above back into (39), one obtains

$$\frac{1}{2} E[\bar{H}(\eta)\eta^T \nabla^2 J(\theta)\eta] = \nabla^2 J(\theta).$$

The claim now follows from (37). □

COROLLARY A.8. *$\| Y(nL) - \nabla^2 J(\theta(n)) \| \to 0$ w.p. 1, as $n \to \infty$ and $\beta \to 0$.*

PROOF. Note that by the triangle inequality,

$$\| Y(nL) - \nabla^2 J(\theta(n)) \| \leq \| Y(nL) - D^2_{\beta,1} J(\theta(n)) \| + \| D^2_{\beta,1} J(\theta(n)) - \nabla^2 J(\theta(n)) \|.$$

The claim now follows from Corollary A.6 and Proposition A.7. □

We now have the following:

LEMMA A.9. *With probability 1, as $n \to \infty$ and $\beta \to 0$,*

$$\| \{P(Y(nL))\}^{-1} - \{P(\nabla^2 J(\theta(n)))\}^{-1} \| \to 0.$$

PROOF. Note that

$$\| \{P(Y(nL))\}^{-1} - \{P(\nabla^2 J(\theta(n)))\}^{-1} \|$$

$$= \| \{P(\nabla^2 J(\theta(n)))\}^{-1}(P(\nabla^2 J(\theta(n)))\{P(Y(nL))\}^{-1} - I) \|$$

$$= \| \{P(\nabla^2 J(\theta(n)))\}^{-1}(P(\nabla^2 J(\theta(n)))\{P(Y(nL))\}^{-1} - P(Y(nL))\{P(Y(nL))\}^{-1}) \|$$

$$= \| \{P(\nabla^2 J(\theta(n)))\}^{-1}(P(\nabla^2 J(\theta(n))) - P(Y(nL)))\{P(Y(nL))\}^{-1}) \|$$

$$\leq \| \{P(\nabla^2 J(\theta(n)))\}^{-1} \| \cdot \| P(\nabla^2 J(\theta(n))) - P(Y(nL)) \| \cdot \| \{P(Y(nL))\}^{-1} \|$$

$$\leq \sup_n \| \{P(\nabla^2 J(\theta(n)))\}^{-1} \| \sup_n \| \{P(Y(nL))\}^{-1} \| \cdot \| P(\nabla^2 J(\theta(n))) - P(Y(nL)) \|$$

$$\to 0 \text{ as } n \to \infty \text{ and } \beta \to 0.$$

This can be seen as follows. Note that the first inequality above follows from the property on induced matrix norms (see Proposition A.12 of Bertsekas and Tsitsiklis [1989]). Also, by Assumption (A1), $\sup_{\theta \in C} \| \nabla^2 J(\theta) \| < \infty$ as C is a closed and bounded set in $\mathcal{R}^N$. Further, from Lemma A.3, $\sup_n \| Y(nL) \| < \infty$ w.p. 1. Also, from Corollary A.8, $\| Y(nL) - \nabla^2 J(\theta(n)) \| \rightarrow 0$ as $n \rightarrow \infty$ and $\beta \rightarrow 0$ w.p. 1. The last then follows from Assumption (A3). □

We now concentrate on the gradient update recursion for $Z(p)$ in Step 1 of algorithm N-SF1 (cf. (29)). By forming suitable martingale sequences as before, similar conclusions as in Lemma A.1 can be drawn. Consider now the following ODEs:

$$\dot{\theta}(t) = 0, \tag{41}$$

$$\dot{Z}(t) = D_{\beta,1} J(\theta) - Z(t). \tag{42}$$

Consider functions $\hat{Z}(t)$ defined according to $\hat{Z}(r(n)) = Z(nL)$ with the maps $t \rightarrow \hat{Z}(t)$ corresponding to continuous linear interpolations on intervals $[r(n), r(n+1)]$. Given $\bar{T} > 0$, define $\{\bar{T}_n\}$ as follows: $\bar{T}_0 = 0$ and for $n \geq 1$, $\bar{T}_n = \min\{r(m) \mid r(m) \geq \bar{T}_{n-1} + \bar{T}\}$. Let $\bar{I}_n = [\bar{T}_n, \bar{T}_{n+1}]$. As before, there exists some integer $q_n > 0$ such that $\bar{T}_n = r(q_n)$. Define also functions $Z^n(t)$, $t \in I_n$, $n \geq 0$, that are obtained as trajectories of the following ODEs:

$$\dot{Z}^n(t) = D_{\beta,1} J(\theta) - Z^n(t), \tag{43}$$

with $Z^n(\bar{T}_n) = \hat{Z}(r(q_n)) = Z(q_n L)$. We have the following analogs of Lemmas A.3 to A.5 and Corollary A.6. (These can be shown in exactly the same way as the preceding results; hence their proofs are not given.)

LEMMA A.10. *The sequence of updates $\{Z(p)\}$ is uniformly bounded with probability 1.*

LEMMA A.11. $\lim_{n \rightarrow \infty} \sup_{t \in \bar{I}_n} \| Z^n(t) - \hat{Z}(t) \| = 0$ *w.p.1.*

LEMMA A.12. *Given $\bar{T}, \gamma > 0$, $((\theta(r(n)+\cdot), Z(r(n)+\cdot))$, is a bounded $(\bar{T}, \gamma)$-perturbation of (42)–(42) for n sufficiently large.*

COROLLARY A.13. $\| Z(nL) - D_{\beta,1} J(\theta(n)) \| \rightarrow 0$ *w.p. 1, as* $n \rightarrow \infty$.

We now have the following analog of Proposition A.7.

PROPOSITION A.14. $\| D_{\beta,1} J(\theta(n)) - \nabla J(\theta(n)) \| \rightarrow 0$ *as* $\beta \rightarrow 0$.

PROOF. Recall that

$$D_{\beta,1} J(\theta(n)) = E\left[\frac{1}{\beta} \eta(n) J(\theta(n) + \beta \eta(n))\right],$$

where $\eta(n) = (\eta_1(n), \ldots, \eta_N(n))^T$ is a vector of independent $N(0,1)$-distributed random variates. Using a Taylor series expansion of $J(\theta(n) + \beta\eta(n))$ around $\theta(n)$,

one has

$$\begin{aligned} D_{\beta,1}J(\theta(n)) &= \frac{1}{\beta}E[\eta(n)(J(\theta(n)) + \beta\eta(n)^T\nabla J(\theta(n)) \\ &\quad + \beta^2\eta(n)^T\nabla^2 J(\theta(n))\eta(n) + o(\beta^2))] \\ &= \frac{1}{\beta}E[\eta(n)J(\theta(n))] + E[\eta(n)\eta(n)^T\nabla J(\theta(n))] \\ &\quad + \beta E[\eta(n)\eta(n)^T\nabla^2 J(\theta(n))\eta(n)] + o(\beta). \end{aligned} \tag{44}$$

As before, it can be easily verified that the first term on the rifht-hand side of (44) is zero. For the second term above, note that

$$E[\eta(n)\eta(n)^T\nabla J(\theta(n))] = E[\eta(n)\eta(n)^T]\nabla J(\theta(n)) = \nabla J(\theta(n)).$$

The last equality follows since $E[\eta(n)\eta(n)^T] = I$ (the identity matrix). A routine calculation also shows that

$$E[\eta(n)\eta(n)^T\nabla^2 J(\theta(n))\eta(n)] = 0.$$

Hence,

$$D_{\beta,1}J(\theta(n)) = \nabla J(\theta(n)) + o(\beta).$$

The claim follows. □

Using Corollary A.13 and Proposition A.14, a simple application of the triangle inequality gives the following:

COROLLARY A.15. *With probability 1, as $n \to \infty$ and $\beta \to 0$,*

$$\| Z(nL) - \nabla J(\theta(n)) \| \to 0.$$

Finally, we consider the slowest timescale recursion. Consider the ODE

$$\dot{\theta} = \tilde{\Gamma}(-\{P(\nabla^2 J(\theta(t)))\}^{-1}\nabla J(\theta(t))), \tag{45}$$

where for any $y \in \mathcal{R}^N$ and a bounded, continuous function $v(\cdot) : \mathcal{R}^N \to \mathcal{R}^N$,

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

Let

$$K \stackrel{\triangle}{=} \{\theta \in C \mid \nabla J(\theta)^T\tilde{\Gamma}(-\{P(\nabla^2 J(\theta))\}^{-1}\nabla J(\theta)) = 0\}.$$

Further, for any set $S \subseteq C$, given $\eta > 0$, $S^\eta \stackrel{\triangle}{=} \{\theta \in C \mid\| \theta - \theta_0 \|\leq \eta, \theta_0 \in S\}$ shall denote the set of all points in C that are in an "η-neighborhood" of the set S. Let $\hat{K}$ denote the set $\{\theta \in C \mid \tilde{\Gamma}(-\{P(\nabla^2 J(\theta))\}^{-1}\nabla J(\theta)) = -\{P(\nabla^2 J(\theta))\}^{-1}\nabla J(\theta)\}$. Let C^o denote the interior of C. Then, one can see that $C^o \subseteq \hat{K}$. We have the following key result.

THEOREM A.16. *Under Assumptions (A1)–(A4), given $\eta > 0$, there exists $\hat{\beta} > 0$, such that, for all $\beta \in (0, \hat{\beta}]$, the sequence $\{\theta(n)\}$ obtained using N-SF1 converges to a point in K^η with probability 1 as $M \to \infty$.*

PROOF. Recall that the parameter update recursion corresponds to

$$\theta\,(n+1) = \Gamma(\theta(n) - a(n)\{P(Y(nL))\}^{-1}Z(nL)). \tag{46}$$

Note that one can rewrite (46) as

$$\begin{aligned}\theta(n+1) \;=\; & \Gamma(\theta(n) - a(n)(\{P(\nabla^2 J(\theta(n)))\}^{-1}\nabla J(\theta(n)) \\ & + (\{P(\nabla^2 J(\theta(n)))\}^{-1} - \{P(Y(nL))\}^{-1})\nabla J(\theta(n)) \\ & + \{P(Y(nL))\}^{-1}(\nabla J(\theta(n)) - Z(nL)))).\end{aligned}$$

Note that as a consequence of Lemma A.9, Corollary A.15, and Assumption (A3), the second and third terms multiplying $a(n)$ above asymptotically vanish as $n \to \infty$ and $\beta \to 0$. One can then view (46) as a noisy Euler discretization of the ODE (45) using a standard approximation argument as on pp. 191–196 of Kushner and Clark [1978]. Note that $J(\theta)$ itself serves as an associated Liapunov function for (45) since (cf. p. 75 of Kushner and Yin [1997]),

$$\frac{dJ(\theta)}{dt} = \nabla J(\theta)^T \dot{\theta} = \nabla J(\theta)^T \tilde{\Gamma}(-\{P(\nabla^2 J(\theta))\}^{-1}\nabla J(\theta)) \leq 0.$$

In particular for $\theta \in \hat{K}$, $\frac{dJ(\theta)}{dt} < 0$ if $\nabla J(\theta) \neq 0$. Now since $J(\theta)$ satisfies Assumption (A1), it is in particular continuous and hence uniformly bounded on the compact set $C \subset \mathcal{R}^N$. Let $\lambda = \sup_\theta J(\theta) < \infty$. Then $\{\theta \mid J(\theta) \leq \lambda\} = C$. It follows from Lasalle's invariance theorem (see Lasalle and Lefschetz [1961])—stated also as Theorem 2.3, p. 76 of Kushner and Yin [1997]—that $\theta(n) \to \theta^*$, for some $\theta^* \in K$, as $n \to \infty$ and $\beta \to 0$. □

Note that for $\theta \in \hat{K} \cap K$, $\nabla J(\theta) = 0$ since $\{P(\nabla^2 J(\theta))\}^{-1}$ is positive definite and symmetric as $P(\nabla^2 J(\theta))$ is by definition of the operator P. Further, as noted in Kushner and Yin [1997] (cf. p. 79), there may be spurious fixed points within K as well. These, however, lie only on the projection set boundary.

Now note that $\bar{K} \equiv \{\theta \mid \nabla J(\theta) = 0\}$ constitutes the set of all Kuhn-Tucker points (not just local minima). However, points that are not local minima shall correspond to unstable equilibria. In principle, the stochastic approximation scheme may get trapped in an unstable equilibrium (cf. Pemantle [1990], Brandiere [1998]). Avoidance of unstable equilibria can be ensured by using additional independent noise. In most practical scenarios, however, stochastic approximation algorithms are known to converge to a stable equilibrium even without additional noise. Our algorithm, by continuity of $J(\cdot)$, converges to an "ϵ-local minimum."

Next, note that we have used Assumption (A1) only to show that the scheme would converge to a local minimum of $J(\theta)$ if β were slowly decreased to zero. The original scheme however is for a given $\beta > 0$. Hence, in the absence of Assumption (A1), consider the following ODE:

$$\dot{\theta} = \tilde{\Gamma}(-\{P(D^2_{\beta,1} J(\theta(t)))\}^{-1} D_{\beta,1} J(\theta(t))). \tag{47}$$

Note that by Lemma 2.1 and the fact that $h(\cdot)$ is Lipschitz continuous, $J(\theta)$ can be seen to be a continuous function. Moreover, since C is a compact set, $J(\theta)$ is also uniformly bounded on C, thereby implying that the ODE (47) is well posed. One can show using the above methods (without assuming (A1))

that the algorithm, for given $\beta > 0$, tracks the stable fixed points of (47) and converges to a θ for which $D_{\beta,1}J(\theta) = 0$. Thus, in our analysis, Assumption (A1) has been used in characterizing the point of convergence as being a local minimum when $\beta \to 0$.

Finally, note that Theorem A.16 merely gives the existence of a $\hat{\beta} > 0$, for given $\epsilon > 0$, such that $\forall \beta \in (0, \hat{\beta}]$, the algorithm converges to an ϵ-local minimum but does not give the precise value of such a $\hat{\beta}$. The variability in the gradient estimates (7)–(8) that we use in G-SF1/N-SF1 and G-SF2/N-SF2, respectively, has been discussed in detail in Styblinski and Tang [1990]. It was observed there that a low value of β greatly increases the variability in estimates of (7), while for (8), this effect is not as significant. We also observe in our analysis that estimates (7) have a higher bias than (8). Similarly, estimates (13) of the Hessian (in N-SF1) have a higher bias compared with estimates (16) (in N-SF2) (see discussion following Proposition A.18). We have seen in our experiments that N-SF2 converges faster than N-SF1 and exhibits fewer inaccuracies compared with the latter as β is made low or high.

A.2 Convergence Analysis of N-SF2

The analysis for this case proceeds along similar lines as N-SF1 with operators $D_{\beta,2}$ and $D^2_{\beta,2}$ being used in place of $D_{\beta,1}$ and $D^2_{\beta,1}$ respectively. We only present here the main results.

PROPOSITION A.17. $\| D^2_{\beta,2}J(\theta(n)) - \nabla^2 J(\theta(n)) \| \to 0$ *as* $\beta \to 0$.

PROOF. Recall that

$$D^2_{\beta,2}J(\theta(n)) = E\left[\frac{1}{2\beta^2}\bar{H}(\eta(n))(J(\theta(n)+\beta\eta(n)) + J(\theta(n)-\beta\eta(n)))\right],$$

where $\eta(n) = (\eta_1(n), \ldots, \eta_N(n))^T$ is a vector of independent $N(0,1)$ random variates. Using suitable Taylor series expansions of $J(\theta(n)+\beta\eta(n))$ and $J(\theta(n)-\beta\eta(n))$ around $\theta(n)$, one gets

$$D^2_{\beta,2}J(\theta(n)) = E\left[\frac{1}{2\beta^2}\bar{H}(\eta(n))(2J(\theta(n)) + \beta^2\eta(n)^T\nabla^2 J(\theta(n))\eta(n) + o(\beta^3))\right].$$

It has been shown in the proof of Proposition A.7 that $E[\bar{H}(\eta(n))J(\theta(n))] = 0$ and $\frac{1}{2}E[\bar{H}(\eta(n))\eta(n)^T\nabla^2 J(\theta(n))\eta(n)] = \nabla^2 J(\theta(n))$, respectively. The claim follows. □

For the gradient estimates of N-SF2, we have the following analog of Proposition A.14.

PROPOSITION A.18. $\| D_{\beta,2}J(\theta(n)) - \nabla J(\theta(n)) \| \to 0$ *as* $\beta \to 0$.

PROOF. Recall that

$$D_{\beta,2}J(\theta(n)) = E\left[\frac{\eta(n)}{2\beta}(J(\theta(n)+\beta\eta(n)) - J(\theta(n)-\beta\eta(n)))\right].$$

Using appropriate Taylor series expansions of $J(\theta(n) + \beta\eta(n))$ and $J(\theta(n) -$

$\beta\eta(n))$, respectively, around the point $\theta(n)$, one gets

$$D_{\beta,2}J(\theta(n)) = E[\eta(n)\eta(n)^T \nabla J(\theta(n))] + o(\beta),$$
$$= \nabla J(\theta(n)) + o(\beta),$$

since $E[\eta(n)\eta(n)^T] = I$. The claim follows. □

It can be seen from the proofs of Propositions A.14 and A.18 that the overall bias in the gradient estimates (8) is much less compared width those in (7). The same is also true of the corresponding Hessian estimates with the two-sided estimates (16) having much less bias compared with the one-sided ones in (13) (see Propositions A.7 and A.17). This is because of the direct cancellation of many of the terms that contribute to the bias in the corresponding Taylor series expansions in the two-sided estimates as opposed to the same terms being mean-zero in the case of estimates in N-SF1. We finally have the following key result whose proof follows along the same lines as that of Theorem A.16.

THEOREM A.19. *Under Assumptions (A1)–(A4), given $\eta > 0$, there exists $\hat{\beta} > 0$, such that for all $\beta \in (0, \hat{\beta}]$, the sequence $\{\theta(n)\}$ obtained using N-SF2 converges to a point in K^η with probability 1 as $M \to \infty$.*

A.3 Convergence Analysis of G-SF2

Note that one can view G-SF2 as being similar to N-SF2 with the difference that one sets here each estimate of the Hessian matrix as I (the identity matrix). Assumption (A3) is thus not needed. Note also that $P(I) = I$ since I is positive definite and symmetric. The analysis proceeds by following a similar sequence of steps as before. The conclusions of Proposition A.18 continue to hold as the gradient estimates in N-SF2 and G-SF2 are the same. Consider now the ODE (tracked by the slowest timescale recursion):

$$\dot{\theta} = \tilde{\Gamma}(-\nabla J(\theta(t)). \tag{48}$$

The set of stable fixed points of this ODE lie within the set $K_1 \stackrel{\triangle}{=} \{\theta \in C \mid \tilde{\Gamma}(-\nabla J(\theta)) = 0\}$. However, note that $K_1 = K$ when $\{P(\nabla^2 J(\theta))\}^{-1}$ is set equal to I here. This can be seen as follows. Suppose $\theta \in K$. Then either $\nabla J(\theta) = 0$ or $\tilde{\Gamma}(-\nabla J(\theta)) = 0$. If the latter holds, then $\theta \in K_1$. On the other hand, if $\nabla J(\theta) = 0$, then $\tilde{\Gamma}(-\nabla J(\theta)) = 0$ as well and $\theta \in K_1$. Next, suppose $\theta \in K_1$. Then clearly $\theta \in K$ as well. Now consider the set $\hat{K}_1 \stackrel{\triangle}{=} \{\theta \in C \mid \tilde{\Gamma}(-\nabla J(\theta)) = -\nabla J(\theta)\}$. It is easy to see that $C^o \subset \hat{K}_1$. Also, one can see that the regular fixed points in K (with the earlier definition of K, namely, with $\{P(\nabla^2 J(\theta))\}^{-1}$ not set to I) also lie in K_1 and vice versa. The only difference between these two sets would only be (if at all) in the spurious fixed points that, however, lie only on the boundary of the projection set. One can now proceed along similar lines as before to show the following

THEOREM A.20. *Under Assumptions (A1), (A2) and (A4), given $\eta > 0$, there exists $\hat{\beta} > 0$, such that for all $\beta \in (0, \hat{\beta}]$, the sequence $\{\theta(n)\}$ obtained using G-SF2 converges to a point in K_1^η with probability 1 as $M \to \infty$.*

A.4 Convergence Analysis of the Jacobi Variants of N-SF1 and N-SF2

The analysis for the Jacobi variants of algorithms N-SF1 and N-SF2 that we used in the numerical experiments also proceeds along similar lines as before. Note, however, that instead of $D^2_{\beta,1}J(\theta)$ and $D^2_{\beta,2}J(\theta)$, one has in this case

$$\bar{D}^2_{\beta,1}J(\theta) = E\left[\frac{1}{\beta^2}\bar{H}_1(\eta)J(\theta+\beta\eta)\right],$$
$$\bar{D}^2_{\beta,2}J(\theta) = E\left[\frac{1}{2\beta^2}\bar{H}_1(\eta)(J(\theta+\beta\eta)+J(\theta-\beta\eta)\right],$$

respectively, as the one- and two-sided estimates of the matrix that is the analog of the Hessian in N-SF1 and N-SF2, respectively. Here $\bar{H}_1(\eta)$ is the matrix

$$\bar{H}_1(\eta) = \begin{bmatrix} ((\eta_1)^2-1) & 0 & \cdots & 0 \\ 0 & ((\eta_2)^2-1) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & ((\eta_N)^2-1) \end{bmatrix}, \tag{49}$$

where $\eta = (\eta_1, \ldots, \eta_N)^T$ is a vector of independent $N(0,1)$ random variates. Let $D^2J(\theta)$ be the matrix

$$D^2J(\theta) = \begin{bmatrix} \nabla_{11}J(\theta) & 0 & \cdots & 0 \\ 0 & \nabla_{22}J(\theta) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \nabla_{NN}J(\theta) \end{bmatrix}. \tag{50}$$

The following directly follows from Propositions A.7 and A.17.

COROLLARY A.21. *For $l = 1, 2$, $\| \bar{D}^2_{\beta,l}J(\theta(n)) - D^2J(\theta(n)) \| \rightarrow 0$, as $\beta \rightarrow 0$.*

Consider now the ODE:

$$\dot{\theta} = \tilde{\Gamma}(-\{P(D^2J(\theta(t)))\}^{-1}\nabla J(\theta(t))). \tag{51}$$

Let

$$K_0 \stackrel{\triangle}{=} \{\theta \in C \mid \nabla J(\theta)^T\tilde{\Gamma}(-\{P(D^2J(\theta))\}^{-1}\nabla J(\theta)) = 0\}.$$

Also, let $\hat{K}_0 \stackrel{\triangle}{=} \{\theta \in C \mid \tilde{\Gamma}(-\{P(D^2J(\theta))\}^{-1}\nabla J(\theta)) = -\{P(D^2J(\theta))\}^{-1}\nabla J(\theta)\}$. One can see that $C^o \subseteq \hat{K}_0$. Note also that, for all $\theta \in \hat{K}_0$, $\nabla J(\theta) = 0$. The discussion following Theorem A.16 is also valid here and it can be seen that the regular fixed points in K would also lie in K_0 and vice versa. The only difference between these two sets (if at all) would only be in spurious fixed points that, however, only lie on the boundary of the projection set. We thus have the following result.

THEOREM A.22. *Under Assumptions (A1)–(A4), given $\eta > 0$, there exists $\hat{\beta} > 0$, such that for all $\beta \in (0, \hat{\beta}]$, the sequence $\{\theta(n)\}$ obtained using either of the Jacobi variants of N-SF1 or N-SF2 converges to a point in K_0^η with probability 1 as $M \rightarrow \infty$.*

ACKNOWLEDGMENTS

The author thanks the Editor-in-Chief Professor Jim Wilson, the Area Editor, the Associate Editor, and both the referees for their many comments that have

helped in significantly enhancing the scope of the contribution and improving the quality of the manuscript. In particular, the author gratefully thanks Reviewer 1 for bringing to his notice the Chin [1997] article, as a result of which algorithms G-SF2 and N-SF2 were developed. The author also thanks Vivek Mishra for helping him with the convergence plots.

REFERENCES

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

BERTSEKAS, D. P. AND TSITSIKLIS, J. N. 1989. *Parallel and Distributed Computation*. Prentice Hall, Englewood Cliffs, NJ.

BHATNAGAR, S. 2005. Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization. *ACM Trans. Model. Comput. Sim. 15,* 1, 74–107.

BHATNAGAR, S. AND BORKAR, V. S. 1998. A two time scale stochastic approximation scheme for simulation based parametric optimization. *Prob. Eng. 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. 2001. 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 WANG, I.-J. 2003. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Trans Model. Comput. Sim. 13,* 2, 180–209.

BORKAR, V. S. 1995. *Probability Theory: An Advanced Course*. Springer Verlag, New York, NY.

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

CASSANDRAS, C. G. 1993. *Discrete Event Systems: Modeling and Performance Analysis*. Aksen Associates, Boston, MA.

CHIN, D. C. 1997. Comparative study of stochastic algorithms for system optimization based on gradient approximation. *IEEE Trans. Sys. Man. Cyber. Part B 27,* 2, 244–249.

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

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

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

KATKOVNIK, V. Y. AND KULCHITSKY, Y. 1972. Convergence of a class of random search algorithms. *Automat. Remote Cont. 8*, 1321–1326.

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

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. *Manage. Sci. 40,* 11, 1562–1578.

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

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

RUBINSTEIN, R. Y. 1981. *Simulation and the Monte Carlo Method*. Wiley, New York, NY.

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

SPALL, J. C. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Auto. Contr. 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. Auto. Contr. 45*, 1839–1853.

SPALL, J. C. 2006. Convergence analysis for feedback- and weighting-based Jacobian estimates in the adaptive simultaneous perturbation algorithm. In *Proceedings of the IEEE Conference on Decision and Control* (San Diego, CA, Dec. 13–15). 5669–5674.

STYBLINSKI, M. A. AND OPALSKI, L. J. 1986. Algorithms and software tools for IC yield optimization based on fundamental fabrication parameters. *IEEE Trans. Comput. Aid. Des. CAD-5,* 1, 79–89.

STYBLINSKI, M. A. AND TANG, T.-S. 1990. Experiments in nonconvex optimization: Stochastic approximation with function smoothing and simulated annealing. *Neur. Netw. 3*, 467–483.

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

Received March 2006; revised January 2007, May 2007; accepted May 2007