Efficient Implementation of Enhanced Adaptive Simultaneous Perturbation Algorithm

Pushpendre Rastogi
Department of Computer Science
The Johns Hopkins University
Baltimore, MD 21218
Email: pushpendre@jhu.edu

Jingyi Zhu
Department of Applied Mathematics and Statistics
The Johns Hopkins University
Baltimore, MD 21218
Email: jingyi.zhu@jhu.edu

James C. Spall
The Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20723
Email: james.spall@jhuapl.edu

***Abstract*—Stochastic approximation (SA) applies in both the gradient-free optimization (Kiefer-Wolfowitz) and the gradient-based setting (Robbins-Monro). The idea of simultaneous perturbation (SP) has been well established. This paper discusses an efficient way of implementing both the adaptive Newton-like SP algorithms and their enhancements (feedback and optimal weighting incorporated), using the Woodbury matrix identity, a.k.a. matrix inversion lemma. Basically, instead of estimating the Hessian matrix directly, this paper deals with the estimation of the inverse of the Hessian matrix. Furthermore, the preconditioning steps, which are required in early iterations to maintain positive-definiteness of the Hessian estimates, are imposed on the Hessian inverse rather than the Hessian itself. Numerical results also demonstrate the superiority of this efficient implementation on Newton-like SP algorithms.**

***Keywords*—Adaptive Estimation; Simultaneous Perturbation Stochastic Approximation (SPSA); Woodbury Matrix Identity**

I. Introduction

Stochastic approximation (SA) has been widely applied in minimization and/or root-finding problems in noisy environment. The Kiefer-Wolfowitz algorithm in [1] can be viewed as a stochastic analogue of steepest descent method, where the stochastic gradient is approximated by a finite difference. The stochastic analogue of Newton-Raphson algorithm is known to provide a nearly-optimal SA algorithms with almost quadratic convergence rate. However in practice, the exact/noisy Hessian information is mostly unavailable. Many algorithms have been proposed to achieve more accurate Hessian approximations. Fabian [2] provides estimates for both the gradient and Hessian information based on finite-difference approximation using noisy function information. Ruppert [3] also presents a finite-difference Hessian approximation scheme using noisy gradient information. Such finite-difference formulations can be notoriously costly in dealing with high-dimensional problems.

To overcome the curse of dimensionality, Spall [4] first introduces the idea of *simultaneous perturbation* (SP), which can be applied in both the gradient-free optimization (Kiefer-Wolfowitz) and the gradient-based setting (Robbins-Monro). Wang and Spall [6] creatively extends SP idea to discrete optimization setting. Later Spall [5] generalizes the SP idea to *adaptive* simultaneous perturbation (ASP) and demonstrated a stochastic analogue of Newton-Raphson algorithm. Spall [7] and [8] incorporate a feedback process and an optimal weighting mechanism into the method in Spall [5], providing an *enhanced* second-order algorithm with more accurate Hessian matrix approximations. Chapter 7 of Bhatnagar et al. [11] extensively explains the framework and convergence analysis for some variants of Newton-type ASP algorithms.

The essence of second-order ASP algorithm, under the minimization setting, is to approximately and efficiently obtain an estimate of the Hessian matrix of the loss function at each iteration using perturbation sequences satisfying certain regularity conditions. The estimate of the Hessian can be obtained solely from noisy loss function evaluations or from noisy gradient evaluations.

Consider the problem of minimizing a differentiable loss function $L(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^p$ with $p \geq 1$. Denote $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L/\partial\boldsymbol{\theta}$. The minimization problem $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ is equivalent to the root finding problem $\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{0}$. Consider the typical case where one only have access to noisy measurements of the function $L(\boldsymbol{\theta})$, say $y(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta})$, or noisy measurements of the gradient $\boldsymbol{g}(\boldsymbol{\theta})$, say $\boldsymbol{Y}(\boldsymbol{\theta}) = \boldsymbol{g}(\boldsymbol{\theta}) + \boldsymbol{e}(\boldsymbol{\theta})$. Throughout this paper, we assume that only noisy function evaluations are available to 2SPSA algorithms, and that 2SG algorithms use both noisy function and gradient evaluations.

This paper presents an efficient way to obtain an accurate inverse of estimate of the Hessian matrix $\overline{\boldsymbol{H}}_k^{-1}$, by eliminating the necessity to solve a linear system of equations in the original algorithms (3) and (8). The basic idea is to maintain and update the value of $\overline{\boldsymbol{H}}_k^{-1}$ by applying the Matrix Inversion Lemma (MIL) in Woodbury [9]:

$$(\boldsymbol{A}+\boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1}+\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}, \tag{1}$$

where $\boldsymbol{A}$, $\boldsymbol{U}$, $\boldsymbol{C}$ and $\boldsymbol{V}$ are of the correct (conformable) sizes and the denoted matrix inverses exist.

Using MIL, we recursively update an estimate of the Hessian inverse rather than the Hessian itself, and pre-

condition the inverse of the Hessian estimate directly. Essentially, first update $\overline{\boldsymbol{H}}_k^{-1}$, and then perform the pre-conditioning on $\overline{\boldsymbol{H}}_k^{-1}$ to obtain $\overline{\overline{\boldsymbol{H}}}_k^{-1}$.

In general, the perturbation sequences $\boldsymbol{\Delta}_k$ and $\tilde{\boldsymbol{\Delta}}_k$ in Section II to IV can be any sequences satisfying the regularity condition C.9 in Spall [8]. A special case of interest as discussed in Section V, is when all components in the perturbation sequences (both $\boldsymbol{\Delta}_k$ and $\tilde{\boldsymbol{\Delta}}_k$) are symmetric Bernoulli distributed, a.k.a. *Rademacher* distributed. Such a perturbation sequence is valid and efficient as shown in Sadegh and Spall [10]. Every component in the symmetric Bernoulli sequence has a 50% chance of being either +1 or –1, such that:

$$\boldsymbol{\Delta}_k = \boldsymbol{\Delta}_k^{-1}, \tilde{\boldsymbol{\Delta}}_k = \tilde{\boldsymbol{\Delta}}_k^{-1}. \tag{2}$$

where for any vector $\boldsymbol{x} \in \mathbb{R}^p \setminus \{\boldsymbol{0}\}$, $\boldsymbol{x}^{-1}$ denotes $[1/x_1, \ldots, 1/x_p]^T$ for the ease of following discussion. Additionally, $\boldsymbol{x}^{-T}$ denotes $[1/x_1, \ldots, 1/x_p]$.

Chapter 5 of a recent textbook Bhatnagar et al. [11] comprehensively covers a deterministic perturbation construction based on the particular structure of Hadamard matrix in Sylvester [12]. Bhatnagar et al. [13] provides some empirical evidences that such a deterministic construction of the perturbation sequence results in better approximation accuracy and speed of convergence than the optimal stochastic generation described in Sadegh and Spall [10]; though the superiority of this deterministic formulation has not been proven theoretically yet. However, all the components of such deterministic Hadamard-matrix-based perturbation are still either +1 or –1, so (2) still holds. In Section V, we shall see the usage of relationship (2).

The efficient implementations of ASP algorithms in this paper aims to reduce the number of computation required in the Hessian inverse. The derivations in Section II to IV are for the coherent rank-two representation of the Hessian update in Section VI. The key of reducing the FLOPs is presented in Section VI: the decomposition of the rank-two update of the Hessian update into a sequential rank-one update of the Hessian inverse. Broadly, the perturbation sequences $\boldsymbol{\Delta}_k$ and $\tilde{\boldsymbol{\Delta}}_k$ in Section II to IV can be any sequences satisfying the regularity condition C.9 in Spall [8], except that the perturbation sequence in the enhanced 2SG case in Section V has to satisfy (2). Section VII further backs up the computational advantage of performing the efficient implementation.

II. Adaptive 2SPSA Algorithm

The adaptive 2SPSA algorithm introduced in Spall [5] employs the following two recursions:

$$\begin{cases} \hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \overline{\overline{\boldsymbol{H}}}_k^{-1} \boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k), & \overline{\overline{\boldsymbol{H}}}_k = \boldsymbol{f}_k(\overline{\boldsymbol{H}}_k), \\ \overline{\boldsymbol{H}}_k = (1-w_k)\overline{\boldsymbol{H}}_{k-1} + w_k \hat{\boldsymbol{H}}_k, & k = 0, 1, \ldots \end{cases} \tag{3}$$

where

$$\begin{cases} w_k = \frac{1}{k+1}, \\ \boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k) = \frac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k)}{2c_k} \boldsymbol{\Delta}_k^{-1}, \\ \hat{\boldsymbol{H}}_k = \frac{1}{2}\left[\frac{\delta \boldsymbol{G}_k}{2c_k}\boldsymbol{\Delta}_k^{-T} + \left(\frac{\delta \boldsymbol{G}_k}{2c_k}\boldsymbol{\Delta}_k^{-T}\right)^T\right], \\ \delta \boldsymbol{G}_k = \boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k) - \boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k), \\ \boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k \pm c_k\boldsymbol{\Delta}_k) = \frac{y(\hat{\boldsymbol{\theta}}_k \pm c_k\boldsymbol{\Delta}_k + \tilde{c}_k\tilde{\boldsymbol{\Delta}}_k) - y(\hat{\boldsymbol{\theta}}_k \pm c_k\boldsymbol{\Delta}_k)}{\tilde{c}_k} \tilde{\boldsymbol{\Delta}}_k^{-1}. \end{cases} \tag{4}$$

a_k, c_k and $\tilde{c}_k$ are all non-negative scalar gain coefficients, $\boldsymbol{\Delta}_k$ and $\tilde{\boldsymbol{\Delta}}_k$ are stochastic perturbation (vector) sequence, and the pre-conditioning function $\boldsymbol{f}_k$ projects an indefinite matrix to some positive definite matrix and is used to maintain the positive-definiteness of $\overline{\overline{\boldsymbol{H}}}_k$.

As mentioned in Section I, we focus on converting the second recursion in algorithm (3) into a recursion on $\overline{\boldsymbol{H}}_k^{-1}$, and directly pre-conditioning $\overline{\boldsymbol{H}}_k^{-1}$ such that

$$\overline{\overline{\boldsymbol{H}}}_k^{-1} = \boldsymbol{f}_k(\overline{\boldsymbol{H}}_k^{-1}). \tag{5}$$

A particular form of $\boldsymbol{f}_k$ suggested in Spall [8] is $\boldsymbol{f}_k : \mathbb{R}^{p\times p} \to \mathbb{R}^{p\times p}$ defined by $\boldsymbol{f}_k(\boldsymbol{H}) = (\boldsymbol{H}^T\boldsymbol{H} + \delta_k \boldsymbol{I}_p)^{1/2}$ where the square root is the (unique) positive definite matrix square root and δ_k is a small positive number. The preconditioning imposed on the Hessian inverse as in (5) generally applies throughout Section II-V.

Denote

$$\begin{aligned} \delta y_k = &[y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k + \tilde{c}_k\tilde{\boldsymbol{\Delta}}_k) - y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k)] \\ &- [y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k + \tilde{c}_k\tilde{\boldsymbol{\Delta}}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k)], \end{aligned} \tag{6}$$

Immediately,

$$\hat{\boldsymbol{H}}_k = \frac{1}{2}\frac{\delta y_k}{2c_k\tilde{c}_k}\left(\tilde{\boldsymbol{\Delta}}_k^{-1}\boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k^{-1}\tilde{\boldsymbol{\Delta}}_k^{-T}\right). \tag{7}$$

Now the updated Hessian can be readily seen as a rank-two update with respect to the current Hessian estimate $\overline{\boldsymbol{H}}_k$:

$$\overline{\boldsymbol{H}}_k = (1-w_k)\overline{\boldsymbol{H}}_{k-1} + \frac{w_k\delta y_k}{4c_k\tilde{c}_k}(\tilde{\boldsymbol{\Delta}}_k^{-1}\boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k^{-1}\tilde{\boldsymbol{\Delta}}_k^{-T}).$$

III. Enhancement for Adaptive 2SPSA Algorithms

Spall [8] improved the adaptive 2SPSA algorithm from Section II by incorporating feedback and optimal weighting mechanisms. The enhanced method uses the following two recursions:

$$\begin{cases} \hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \overline{\overline{\boldsymbol{H}}}_k^{-1} \boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k), & \overline{\overline{\boldsymbol{H}}}_k = \boldsymbol{f}_k(\overline{\boldsymbol{H}}_k), \\ \overline{\boldsymbol{H}}_k = (1-w_k)\overline{\boldsymbol{H}}_{k-1} + w_k(\hat{\boldsymbol{H}}_k - \hat{\boldsymbol{\Psi}}_k), & k = 0, 1, \ldots \end{cases} \tag{8}$$

where $\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, $\hat{\boldsymbol{H}}_k$, $\delta\boldsymbol{G}_k$, and $\boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k \pm c_k\boldsymbol{\Delta}_k)$ are defined in (4), and notation in (6) and (7) in Section II apply. The optimal weighting parameter, which minimizes the asymptotic variances of the elements in $\hat{\boldsymbol{H}}_k$, is proven to be:

$$w_k = \frac{\tilde{c}_k^2 c_k^2}{\sum_{i=0}^k \tilde{c}_i^2 c_i^2}. \tag{9}$$

Eq. (3.8) in Spall [8] provides a symmetrized feedback term $\hat{\boldsymbol{\Psi}}_k$:

$$\hat{\boldsymbol{\Psi}}_k = (\hat{\boldsymbol{\Phi}}_k + \hat{\boldsymbol{\Phi}}_k^T)/2, \tag{10}$$

where the pre-symmetrized form of $\hat{\boldsymbol{\Psi}}_k$:

$$\hat{\boldsymbol{\Phi}}_k = \tilde{\boldsymbol{D}}_k^T \overline{\boldsymbol{H}}_{k-1} \boldsymbol{D}_k + \tilde{\boldsymbol{D}}_k^T \overline{\boldsymbol{H}}_{k-1} + \overline{\boldsymbol{H}}_{k-1} \boldsymbol{D}_k. \tag{11}$$

with $\boldsymbol{D}_k = \boldsymbol{\Delta}_k \boldsymbol{\Delta}_k^{-T} - \boldsymbol{I}_p$, $\tilde{\boldsymbol{D}}_k = \tilde{\boldsymbol{\Delta}}_k \tilde{\boldsymbol{\Delta}}_k^{-T} - \boldsymbol{I}_p$.

Collecting terms in $\hat{\boldsymbol{\Phi}}_k$ gives:

$$\begin{aligned} &\tilde{\boldsymbol{D}}_k^T \overline{\boldsymbol{H}}_{k-1} \boldsymbol{D}_k + \tilde{\boldsymbol{D}}_k^T \overline{\boldsymbol{H}}_{k-1} + \overline{\boldsymbol{H}}_{k-1} \boldsymbol{D}_k \\ &= \tilde{\boldsymbol{\Delta}}_k^{-1} \tilde{\boldsymbol{\Delta}}_k^T \overline{\boldsymbol{H}}_{k-1} \boldsymbol{\Delta}_k \boldsymbol{\Delta}_k^{-T} - \overline{\boldsymbol{H}}_{k-1}. \end{aligned}$$

Now we note that the symmetry of previous update $\overline{\boldsymbol{H}}_{k-1}$ guarantees that $\boldsymbol{\Delta}_k^T \overline{\boldsymbol{H}}_{k-1} \tilde{\boldsymbol{\Delta}}_k = \tilde{\boldsymbol{\Delta}}_k^T \overline{\boldsymbol{H}}_{k-1} \boldsymbol{\Delta}_k$. Denote

$$b_k = \frac{w_k}{2} \left(\frac{\delta y_k}{2 c_k \tilde{c}_k} - \boldsymbol{\Delta}_k^T \overline{\boldsymbol{H}}_{k-1} \tilde{\boldsymbol{\Delta}}_k \right). \tag{12}$$

Now we can obtain $\overline{\boldsymbol{H}}_k$ as a rank-two update of $\overline{\boldsymbol{H}}_{k-1}$:

$$\begin{aligned} \overline{\boldsymbol{H}}_k &= (1 - w_k) \overline{\boldsymbol{H}}_{k-1} + w_k (\hat{\boldsymbol{H}}_k - \hat{\boldsymbol{\Psi}}_k) \\ &= \overline{\boldsymbol{H}}_{k-1} + \frac{w_k}{2} \left(\frac{\delta y_k}{2 c_k \tilde{c}_k} \tilde{\boldsymbol{\Delta}}_k^{-1} \boldsymbol{\Delta}_k^{-T} - \tilde{\boldsymbol{\Delta}}_k^{-1} \tilde{\boldsymbol{\Delta}}_k^T \overline{\boldsymbol{H}}_{k-1} \boldsymbol{\Delta}_k \boldsymbol{\Delta}_k^{-T} \right) \\ &\quad + \frac{w_k}{2} \left(\frac{\delta y_k}{2 c_k \tilde{c}_k} \boldsymbol{\Delta}_k^{-1} \tilde{\boldsymbol{\Delta}}_k^{-T} - \boldsymbol{\Delta}_k^{-1} \boldsymbol{\Delta}_k^T \overline{\boldsymbol{H}}_{k-1} \tilde{\boldsymbol{\Delta}}_k \tilde{\boldsymbol{\Delta}}_k^{-T} \right) \\ &= \overline{\boldsymbol{H}}_{k-1} + b_k (\tilde{\boldsymbol{\Delta}}_k^{-1} \boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k^{-1} \tilde{\boldsymbol{\Delta}}_k^{-T}). \end{aligned}$$

IV. Adaptive 2SG Algorithm

Now we have access to the noisy measurement of the gradient information, $\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, $\boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\Delta}_k)$ and $\boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\Delta}_k)$ at each iteration k. An adaptive 2SG algorithm introduced in Spall [5] has the recursion as algorithm (3) with

$$\begin{cases} w_k = \dfrac{1}{k+1}, \\ \delta \boldsymbol{G}_k = \boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\Delta}_k) - \boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\Delta}_k), \\ \hat{\boldsymbol{H}}_k = \dfrac{1}{2} \left[\dfrac{\delta \boldsymbol{G}_k}{2 c_k} \boldsymbol{\Delta}_k^{-T} + \left(\dfrac{\delta \boldsymbol{G}_k}{2 c_k} \boldsymbol{\Delta}_k^{-T} \right)^T \right]. \end{cases} \tag{13}$$

This leads to the following rank two update with respect to the current Hessian estimate $\overline{\boldsymbol{H}}_k$:

$$\overline{\boldsymbol{H}}_k = (1 - w_k) \overline{\boldsymbol{H}}_{k-1} + \frac{w_k}{4 c_k} ((\delta \boldsymbol{G}_k) \boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k^{-1} (\delta \boldsymbol{G}_k)^T).$$

V. Enhancement for Adaptive 2SG Algorithms

The enhancement on adaptive 2SG algorithm can be achieved analogously to that on adaptive 2SPSA algorithm, by incorporating a feedback mechanism and optimal weighting in (9). Same as Section IV, enhanced 2SG algorithms have access to the noisy measurement of the gradient information, $\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, $\boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\Delta}_k)$ and $\boldsymbol{G}_k^{(1)}(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\Delta}_k)$ at each iteration k.

Eq. (3.12) in Spall [8] provides a symmetrized feedback term $\hat{\boldsymbol{\Psi}}_k$, which can be rewritten as the following using relationship (2):

$$\hat{\boldsymbol{\Psi}}_k = (\overline{\boldsymbol{H}}_{k-1} \boldsymbol{D}_k + \boldsymbol{D}_k \overline{\boldsymbol{H}}_{k-1})/2, \tag{14}$$

where $\boldsymbol{D}_k = \boldsymbol{\Delta}_k \boldsymbol{\Delta}_k^{-T} - \boldsymbol{I}_p$.

If we only use the symmetric Bernoulli (Rademacher) perturbation sequence as suggested in Sadegh and Spall [10], then $\boldsymbol{D}_k$ is symmetric. Meanwhile, $\hat{\boldsymbol{H}}_k$ and $\hat{\boldsymbol{\Psi}}_k$ can simplified as following:

$$\begin{cases} \hat{\boldsymbol{H}}_k &= \dfrac{1}{2} \dfrac{1}{2 c_k} \left((\delta \boldsymbol{G}_k) \boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k (\delta \boldsymbol{G}_k)^T \right), \\ \hat{\boldsymbol{\Psi}}_k &= \dfrac{1}{2} \left(\overline{\boldsymbol{H}}_{k-1} \boldsymbol{\Delta}_k \boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k \boldsymbol{\Delta}_k^T \overline{\boldsymbol{H}}_{k-1} - 2 \overline{\boldsymbol{H}}_{k-1} \right). \end{cases} \tag{15}$$

Denote

$$\boldsymbol{u}_k = \frac{1}{2 c_k} \delta \boldsymbol{G}_k - \overline{\boldsymbol{H}}_{k-1} \boldsymbol{\Delta}_k. \tag{16}$$

Using the symmetry of update $\overline{\boldsymbol{H}}_{k-1}$, we obtain a rank-two update with respect to the current Hessian estimate $\overline{\boldsymbol{H}}_k$:

$$\overline{\boldsymbol{H}}_k = \overline{\boldsymbol{H}}_{k-1} + \frac{w_k}{2} (\boldsymbol{u}_k \boldsymbol{\Delta}_k^{-T} + \boldsymbol{\Delta}_k^{-1} \boldsymbol{u}_k^T).$$

VI. Efficient Update for Hessian Inverse

As described in Section II–V, the Hessian updates in efficient implementations can be described as rank-two updates in all variants of adaptive simultaneous perturbation algorithms. The Hessian update (either with or without feedback term) can be generalized as:

$$\overline{\boldsymbol{H}}_k = d_k \overline{\boldsymbol{H}}_{k-1} + b_k (\boldsymbol{u}_k \boldsymbol{v}_k^T + \boldsymbol{v}_k \boldsymbol{u}_k^T). \tag{17}$$

where the detailed expressions for scalar terms d_k and b_k, and vectors $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$, are listed in Table I for all four cases. Note that condition (2) is only required for the enhanced 2SG case marked by *.

The procedure in eq. (17) requires at least $4p^2 + p$ operations (scalar addition and multiplication). Recursion (17) has to be updated in the original algorithms (3) and (8) for all four cases. However, in the efficient implementation setting, it needs to be updated only for enhanced 2SPSA in Section III and enhanced 2SG in Section V, as evidenced in Table I.

Recognizing the rank-two structure of the Hessian updates in eq. (17) leads to an efficient update recursion for the Hessian inverse by applying the MIL (1). First compute

$$\begin{cases} \tilde{\boldsymbol{u}}_k = \sqrt{\dfrac{|\boldsymbol{v}_k|}{2|\boldsymbol{u}_k|}} \left(\boldsymbol{u}_k + \dfrac{|\boldsymbol{u}_k|}{|\boldsymbol{v}_k|} \boldsymbol{v}_k \right) \\ \tilde{\boldsymbol{v}}_k = \sqrt{\dfrac{|\boldsymbol{v}_k|}{2|\boldsymbol{u}_k|}} \left(\boldsymbol{u}_k - \dfrac{|\boldsymbol{u}_k|}{|\boldsymbol{v}_k|} \boldsymbol{v}_k \right) \end{cases} \tag{18}$$

such that

$$\boldsymbol{u}_k \boldsymbol{v}_k^T + \boldsymbol{v}_k \boldsymbol{u}_k^T = \tilde{\boldsymbol{u}}_k \tilde{\boldsymbol{u}}_k^T - \tilde{\boldsymbol{v}}_k \tilde{\boldsymbol{v}}_k^T.$$

Although the computation of $\tilde{\boldsymbol{u}}_k, \tilde{\boldsymbol{v}}_k$ requires $6p$ operations (addition, multiplication and square root), it reduces the number of required matrix vector multiplications from 4 down to 2, and it maintains symmetry of updates

Algorithms	d_k	b_k	$\boldsymbol{u}_k$	$\boldsymbol{v}_k$
Adaptive 2SPSA	$1-w_k$	$\frac{w_k \delta y_k}{4c_k \tilde{c}_k}$	$\tilde{\boldsymbol{\Delta}}_k^{-1}$	
Enhanced 2SPSA	1	$\frac{w_k}{2}(\frac{\delta y_k}{2c_k\tilde{c}_k} - \boldsymbol{\Delta}_k^T \overline{\boldsymbol{H}}_{k-1}\tilde{\boldsymbol{\Delta}}_k)$	$\tilde{\boldsymbol{\Delta}}_k^{-1}$	$\boldsymbol{\Delta}_k^{-1}$
Adaptive 2SG	$1-w_k$	$\frac{w_k}{4c_k}$	$\delta \boldsymbol{G}_k$	
Enhanced 2SG *	1	$\frac{w_k}{2}$	$\frac{1}{2c_k}(\delta \boldsymbol{G}_k) - \overline{\boldsymbol{H}}_{k-1}\boldsymbol{\Delta}_k$	

TABLE I: Expressions for Terms in eq. (17).
In original algorithms (3) and (8), eq. (17) needs to be updated for all four algorithms.
In efficient implementation, eq. (17) needs to be updated only for Enhanced 2SPSA and 2SG *.
Enhanced 2SG * requires symmetric Bernoulli perturbation sequences.

even under numerical approximations. Here is the update recursion of Hessian inverse $\overline{\boldsymbol{H}}_k^{-1}$:

$$\begin{cases} \boldsymbol{B}_k^{-1} & = d_k^{-1}\overline{\boldsymbol{H}}_{k-1}^{-1} - \frac{d_k^{-2}}{b_k^{-1}+\tilde{\boldsymbol{u}}_k^T\overline{\boldsymbol{H}}_{k-1}^{-1}\tilde{\boldsymbol{u}}_k}\overline{\boldsymbol{H}}_{k-1}^{-1}\tilde{\boldsymbol{u}}_k\tilde{\boldsymbol{u}}_k^T\overline{\boldsymbol{H}}_{k-1}^{-1}, \\ \overline{\boldsymbol{H}}_k^{-1} & = \boldsymbol{B}_k^{-1} + \frac{1}{b_k^{-1}-\tilde{\boldsymbol{v}}_k^T\boldsymbol{B}_k^{-1}\tilde{\boldsymbol{v}}_k}\boldsymbol{B}_k^{-1}\tilde{\boldsymbol{v}}_k\tilde{\boldsymbol{v}}_k^T\boldsymbol{B}_k^{-1}. \end{cases} \tag{19}$$

where $\boldsymbol{B}_k = d_k\overline{\boldsymbol{H}}_{k-1} + b_k\tilde{\boldsymbol{u}}_k\tilde{\boldsymbol{u}}_k^T$.

Algorithms	FLOPS
Adaptive 2SPSA	$9p^2+10p$
Enhanced 2SPSA	$15p^2+13p$
Adaptive 2SG	$9p^2+10p$
Enhanced 2SG*	$15p^2+13p$

TABLE II: FLOPS required in eq. (18)–(19)
Enhanced 2SG* requires symmetric Bernoulli perturbation sequences.

The rank-two update (two sequential rank-1 updates) in equation (19) requires $9p^2+4p$ operations per iteration. The total number of operations required to perform all four efficient implementations from Section II to V are listed in Table II. Now we can proceed to do the FLOPs comparison between the efficient implementation (18)–(19) to the Hessian (inverse) update (including computation of $\overline{\overline{\boldsymbol{H}}}_k^{-1}\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, excluding preconditioning step) in original algorithms (3) and (8).

In algorithm (3) and (8), we need to compute $\overline{\overline{\boldsymbol{H}}}_k^{-1}\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, where $\overline{\overline{\boldsymbol{H}}}_k$ is expected to be symmetric positive definite due to the symmetric update and the preconditioning step $\boldsymbol{f}_k$. The fastest algorithm of handling linear system $\overline{\overline{\boldsymbol{H}}}_k\boldsymbol{x} = \boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$ is Cholesky factorization. In fact, when solving system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with the mldivide function (equivalently linsolve), MATLAB will switch to Cholesky solver when $\boldsymbol{A}$ is positive definite. This can be evidenced by the command setup below that produces information about choice of algorithm based on matrix structure, and about storage allocation.

```
spparms('spumoni',1);
```

Page 63 in Hämmerlin and Hoffmann [14] provides the complexity of Cholesky decomposition: $(p^3+3p^2-p)/3$. Page 602 in Datta [15] states that after the Cholesky factorization (system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ replaced by $\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{x} = \boldsymbol{b}$), the mldivide (backslash operator) recognizes triangular systems. So the converted system is then solved by backward and forward substitution as $\boldsymbol{x} = \boldsymbol{R}\backslash(\boldsymbol{R}^T\backslash\boldsymbol{b})$ with $2p^2$ FLOPs. So the total FLOPs for Hessian update as shown in eq. (17) Hessian (inverse) update (including computation of $\overline{\overline{\boldsymbol{H}}}_k^{-1}\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, excluding preconditioning step) in the original algorithms (3) and (8) is

$$4p^2+p+\frac{1}{3}(p^3+3p^2-p)+2p^2 = \frac{1}{3}p^3+7p^2+\frac{2}{3}p \tag{20}$$

When $p > 9$ in applying basic *adaptive* algorithms, or when $p > 25$ in applying *enhanced* algorithms, the corresponding number of operations is fewer than $p^3/3 + 7p^2 + 2p/3$ as required in original algorithms (3) and (8).

Assume that the time taken for every computation (addition, multiplication, etc.) is the same. Then for the adaptive 2SPSA and 2SG setting, the theoretical ratio of time taken in the efficient implementation over the time taken in Hessian (inverse) update (including computation of $\overline{\overline{\boldsymbol{H}}}_k^{-1}\boldsymbol{G}_k(\hat{\boldsymbol{\theta}}_k)$, excluding preconditioning step) in the algorithms (3) and (8) is at least:

$$\frac{9p^2+10p}{p^3/3+7p^2+2p/3} = O\left(\frac{1}{p}\right). \tag{21}$$

Similarly, for the enhanced 2SPSA and 2SG setting, the theoretical ratio is

$$\frac{15p^2+13p}{p^3/3+7p^2+2p/3} = O\left(\frac{1}{p}\right). \tag{22}$$

The efficient implementation on the Hessian inverse update (18)–(19) is faster than the original setup, in the sense that the FLOPs required in computing the Hessian inverse is reduced from $O(p^3)$ to $O(p^2)$.

VII. Numerical Study

We now present an empirical study of the runtime performance of the proposed updates for the case of adaptive 2SPSA in Section II and enhanced 2SPSA in Section III, on the skewed-quartic function in Spall [8].

$$L(\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{\theta} + 0.1\sum_{i=1}^{p}(\boldsymbol{B}\boldsymbol{\theta})_i^3 + 0.01\sum_{i=1}^{p}(\boldsymbol{B}\boldsymbol{\theta})_i^4.$$

where the dimensionality of the vectors $p = 15$ in this numerical study. $\boldsymbol{B}$ is such that $p\boldsymbol{B}$ is an upper triangular matrix of all 1's. Easily we can derive

$$\begin{cases} \boldsymbol{g}(\theta) & = \boldsymbol{B}^T \left(2\boldsymbol{B}\theta + 0.3\sum_{i=1}^{p}(\boldsymbol{B}\theta)_i^2 + 0.04\sum_{i=1}^{p}(\boldsymbol{B}\theta)_i^3 \right), \\ \boldsymbol{H}(\theta) & = \boldsymbol{B}^T \left[\mathrm{diag}\left(2 + 0.6 * \boldsymbol{B}\theta + 0.12\sum_{i=1}^{p}(\boldsymbol{B}\theta)_i^2 \right) \right] \boldsymbol{B}. \end{cases} \tag{23}$$

It can be shown that that $L(\theta)$ is a strictly convex function, and its minimizer is $\theta^* = \mathbf{0}$, which gives $L(\theta^*) = 0$.

Both adaptive 2SPSA (A2SPSA) algorithm and its enhancement (E2SPSA) require a choice of the gain sequence a_k, the averaging sequence w_k, the perturbation size sequences $c_k, \tilde{c}_k$, the preconditioning sequence δ_k and the number of iterations N. We set the budget to be $20,000$ noisy loss evaluations per iteration. Accordingly, set $n = 5000$, $a_k = a/(A+k+1)^\alpha$, $w_k = w/(k+1)^d$, $c_k = \tilde{c}_k = c/(k+1)^\gamma$ and $\delta_k = 10^{-8}e^{-k}$ where the values of $a, A, w, d, \alpha, \gamma$ are determined following the practical guidelines in Spall [5] and/or via tuning process. Particularly, $a = 1, w = 0.1, d = 0.501, \alpha = 1, A = 50, \gamma = 1/6$. Additionally, the heuristic of bounding (discribed in Section II in [5]) is applied: the iterate at each step is restricted within a ball (in Euclidean norm), centered at the current estimate, with a radius set to be 10.

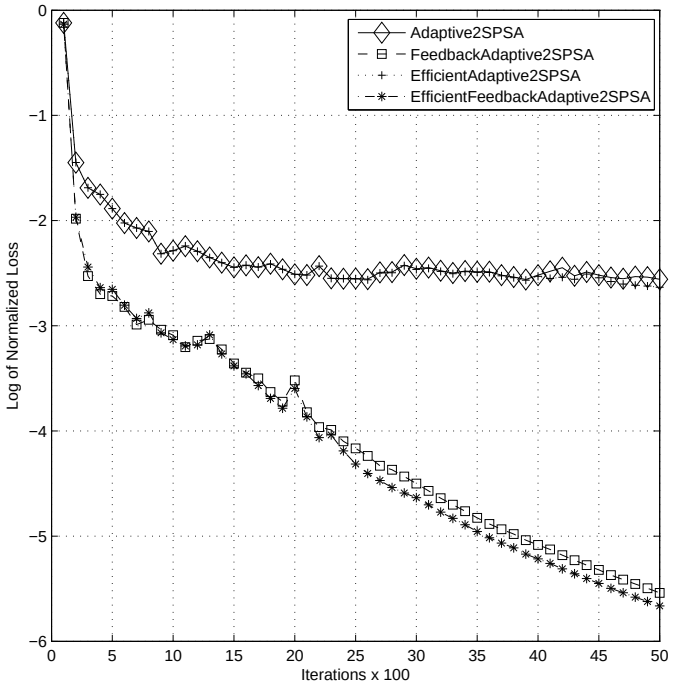

Fig. 1

To compare the original 2SPSA formulation, where the computation of $\overline{\overline{\boldsymbol{H}}}_k^{-1}\boldsymbol{G}_k(\hat{\theta}_k)$, with its more efficient implementation, we first plot the normalized loss $[L(\hat{\theta}_k) - L(\theta^*)]/[L(\hat{\theta}_0) - L(\theta^*)]$ against iteration k averaging over 20 simulation runs. Note that both versions of the algorithms use the *same* random seed and are essentially performing the same iterations. Therefore we would expect that the normalized loss sequences should be close to identical. This is evidenced by Figure 1.

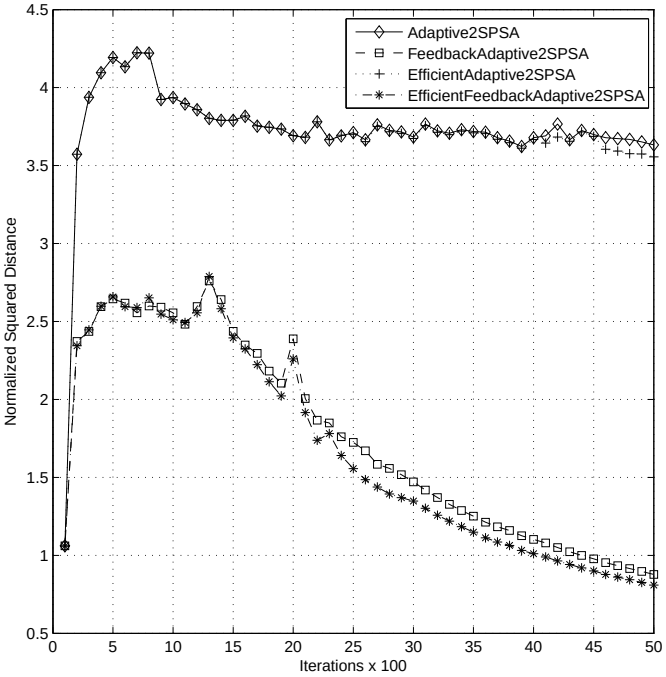

Fig. 2

Figure 2 plots the Euclidean distance between $\hat{\theta}_k$ and θ^* against iteration k averaging over 20 simulation runs.

Below is the table showing the time consumed in each step for $p = 15$ case: Hessian (inverse) update, pre-conditioning on Hessian (inverse), blocking step, and loss function evaluation, for A2SPSA and E2SPSA and their efficient implementations, efficient adaptive 2SPSA (EA2SPSA) and efficient enhanced 2SPSA (EE2SPSA).

Algorithm	Hessian Update	Pre-Conditioning	Blocking	Evaluation	Total
A2SPSA	0.363	1.836	0.431	0.699	3.330
EA2SPSA	0.297	1.702	0.426	0.698	3.123
E2SPSA	0.432	3.079	0.432	0.700	4.642
EEA2SPSA	0.334	1.738	0.435	0.712	3.220

TABLE III: Time Consumed in Each Procedure for $p = 15$
The unit of time is second.

Only consider the time consumed in the Hessian update recursion at each iteration. The second column in Table III shows that, the EA2SPSA only takes 81.82% of the time taken in A2SPSA, and EE2SPSA only takes 77.31% of that in E2SPSA.

The *efficient* implementation can save more time with the original algorithms (3) and (8), as the dimension p goes up. To validate the relationship (21)–(22), we run the algorithms, using the same parameters tuned for dimension $p = 15$, on different dimension $p = 15, 30, 45, 60, 120$ and 240.

Dimension	A2SPSA	EA2SPA	Ratio
15	0.363	0.297	0.8182
30	0.525	0.369	0.7029
45	0.790	0.487	0.6165
60	1.094	0.654	0.5978
120	3.575	1.844	0.5158
240	1.881	0.942	0.5008

TABLE IV: Time Consumed in Hessian Update Procedure for Different Dimension p

Ratio is time taken in EA2SPSA over time taken in A2SPSA.

Dimension	E2SPSA	EE2SPA	Ratio
15	0.432	0.334	0.7731
30	0.565	0.398	0.7044
45	0.844	0.519	0.6149
60	1.138	0.709	0.6230
120	3.517	1.899	0.5399
240	1.984	0.935	0.4713

TABLE V: Time Consumed in Hessian Update Procedure for Different Dimension p

Ratio is time taken in EE2SPSA over time taken in E2SPSA.

From tables IV–V, we can envision the trend that the time consumed in the Hessian update procedure, where the sequential rank-one update (19) makes a difference, is lesser in the efficient implementations than in the original algorithms (3) and (8). Though the ratio of time taken in efficient implementation over that in original setup is indeed decreasing as the dimension goes up, the ratio is not decreasing as fast as (21)–(22) predict. The slower rate results from the efficiency of mldivide function (equivalently linsolve). The dimensions of our test matrices are far too small to manifest the asymptotic $p^3/3$ running time. Furthermore, MATLAB will typically be making use of parallel routines for computing the Cholesky factorization, which saves more time in computation.

In short, this numerical example does show the superiority of the efficient implementation of (18)–(19), in terms of the running time and accuracy (though small as shown in Fig 2). The asymptotic ratio (21)–(22) has not been fully exploited in this small-dimensional example, though it is theoretically valid.

VIII. Conclusion and Future Work

In this paper, we discuss an efficient implementation of four Newton-like SP algorithms: adaptive 2SPSA, enhanced 2SPSA, adaptive 2SG, and enhanced 2SG. The efficiency, in terms of both time taken and the steps required to achieved a certain numerical accuracy, is attained by a unified sequential (rank-two) update using Woodbury matrix identity. Particularly for the case of enhanced 2SG*, the usage of symmetric Bernoulli perturbation sequence is required.

As discovered from practical experience, the time taken in the stochastic optimization (minimization) problem is dominated by the noisy function/gradient evaluation and the preconditioning part. We are gathering ideas in speeding up the preconditioning step in the algorithm, so as to magnify the contribution of this paper in streamlining the Hessian inverse update.

Acknowledgment

The first author is sponsored by the Defense Advanced Research Projects Agency (DARPA) under the Deep Exploration and Filtering of Text (DEFT) Program (Agreement Number: FA8750-13-2-001). Both the second and the third author receive support from the Office of Naval Research (via Navy contract N00024-13-D6400).

References

[1] Kiefer, J., and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function.*The Annals of Mathematical Statistics, 23(3)*, 462-466.

[2] Fabian, V. (1971). Stochastic approximation, *Optimizing Methods in Statistics*, J.S. Rustigi, Ed. New York: Academic Press, pp. 439-470.

[3] Ruppert, D. (1985). A Newton-Raphson version of the multivariate Robbins-Monro procedure. *The Annals of Statistics*, 236-245.

[4] Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3), 332-341.

[5] Spall, J. C. (2000). Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10), 1839-1853.

[6] Wang, Q., and Spall, J. C. (2011). Discrete simultaneous perturbation stochastic approximation on loss function with noisy measurements. In *Proceedings of the American Control Conference, San Francisco*, Vol. 29, pp. 4520-4525.

[7] Spall, J. C. (2007). Feedback and Weighting Mechanisms for Improving Jacobian Estimates in the Adaptive Simultaneous Perturbation Algorithm. In *41st Annual Conference on Information Sciences and Systems* (pp. 35-40). IEEE.

[8] Spall, J. C. (2009). Feedback and weighting mechanisms for improving Jacobian estimates in the adaptive simultaneous perturbation algorithm. *IEEE Transactions on Automatic Control*, 54(6), 1216-1229.

[9] Woodbury, M. A. (1950). Inverting Modified Matrices, Memorandum Rept. 42. *Statistical Research Group, Princeton University, Princeton, NJ, 316*.

[10] Sadegh, P. and Spall, J. C. (1998). Optimal random perturbations for stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, vol. 43, pp. 1480-1484 (correction to references: vol. 44, pp. 231-232).

[11] Bhatnagar, S., Prasad, H. L. and Prashanth, L. A. (2013). *Stochastic recursive algorithms for optimization: simultaneous perturbation methods*, Springer.

[12] Sylvester, J. J. (1867). LX. Thoughts on inverse orthogonal matrices, simultaneous signsuccessions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 34(232), pp. 461-475.

[13] Bhatnagar, S., Fu, M.C., Marcus, S.I. and Wang, I., 2003. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS), 13(2)*, pp.180-209.

[14] Hämmerlin, G., and Hoffmann, K. H. (2012). *Numerical mathematics*. Springer Science & Business Media.

[15] Datta, K. B. (2008). *Matrix And Linear Algebra, Edition 2: AIDED WITH MATLAB*. PHI Learning Pvt. Ltd..