An Overview of the Simultaneous Perturbation Method for Efficient Optimization

James C. Spall

Multivariate stochastic optimization plays a major role in the analysis and control of many engineering systems. In almost all real-world optimization problems, it is necessary to use a mathematical algorithm that iteratively seeks out the solution because an analytical (closed-form) solution is rarely available. In this spirit, the "simultaneous perturbation stochastic approximation (SPSA)" method for difficult multivariate optimization problems has been developed. SPSA has recently attracted considerable international attention in areas such as statistical parameter estimation, feedback control, simulation-based optimization, signal and image processing, and experimental design. The essential feature of SPSA—which accounts for its power and relative ease of implementation—is the underlying gradient approximation that requires only two measurements of the objective function regardless of the dimension of the optimization problem. This feature allows for a significant decrease in the cost of optimization, especially in problems with a large number of variables to be optimized.
(Keywords: Gradient approximation, Multivariate optimization, Simulation-based optimization, Simultaneous perturbation stochastic approximation, SPSA, Stochastic optimization.)

INTRODUCTION

This article is an introduction to the simultaneous perturbation stochastic approximation (SPSA) algorithm for stochastic optimization of multivariate systems. Optimization algorithms play a critical role in the design, analysis, and control of most engineering systems and are in widespread use in the work of APL and other organizations:

The future, in fact, will be full of [optimization] algorithms. They are becoming part of almost everything. They are moving up the complexity chain to make entire companies more efficient. They also are moving down the chain as computers spread. (USA Today, 31 Dec 1997)

Before presenting the SPSA algorithm, we provide some general background on the stochastic optimization context of interest here.

The mathematical representation of most optimization problems is the minimization (or maximization) of some scalar-valued objective function with respect to a vector of adjustable parameters. The optimization algorithm is a step-by-step procedure for changing the adjustable parameters from some initial guess (or set of guesses) to a value that offers an improvement in the objective function. Figure 1 depicts this process for a very simple case of only two variables, θ_1 and θ_2, where our objective function is a loss function to be minimized (without loss of generality, we will discuss optimization in the context of minimization because a maximization problem can be trivially converted to a minimization problem by changing the sign of the objective function). Most real-world problems would have many more variables. The illustration in Fig. 1 is typical of a stochastic optimization setting with noisy input information because the loss function value does not uniformly decrease as the iteration process proceeds (note the temporary increase in the loss value in the third step of the algorithm). Many optimization algorithms have been developed that assume a deterministic setting and that assume information is available on the gradient vector associated with the loss function (i.e., the gradient of the loss function with respect to the parameters being optimized). However, there has been a growing interest in recursive optimization algorithms that do not depend on direct gradient information or measurements. Rather, these algorithms are based on an approximation to the gradient formed from (generally noisy) measurements of the loss function. This interest has been motivated, for example, by problems in the adaptive control and statistical identification of complex systems, the optimization of processes by large Monte Carlo simulations, the training of recurrent neural networks, the recovery of images from noisy sensor data, and the design of complex queuing and discrete-event systems. This article focuses on the case where such an approximation is going to be used as a result of direct gradient information not being readily available.

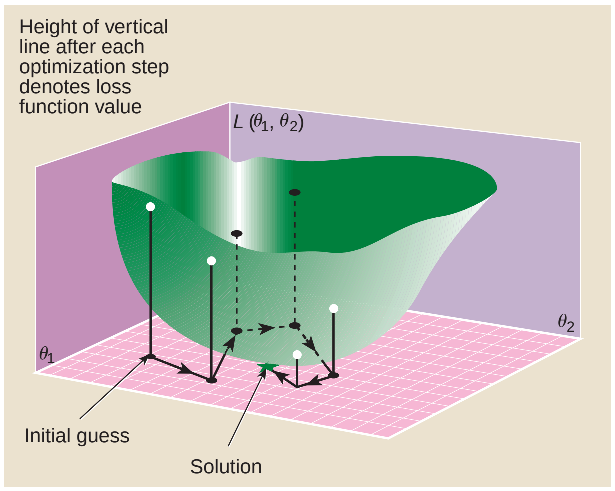
Height of vertical line after each optimization step denotes loss function value
$L(\theta_1, \theta_2)$
θ_1
θ_2
Initial guess
Solution

Figure 1. Example of stochastic optimization algorithm minimizing loss function $L(\theta_1, \theta_2)$.

Overall, gradient-free stochastic algorithms exhibit convergence properties similar to the gradient-based stochastic algorithms [e.g., Robbins-Monro[1] stochastic approximation (R-M SA)] while requiring only loss function measurements. A main advantage of such algorithms is that they do not require the detailed knowledge of the functional relationship between the parameters being adjusted (optimized) and the loss function being minimized that is required in gradient-based algorithms. Such a relationship can be notoriously difficult to develop in some areas (e.g., nonlinear feedback controller design), whereas in other areas (such as Monte Carlo optimization or recursive statistical parameter estimation), there may be large computational savings in calculating a loss function relative to that required in calculating a gradient.

Let us elaborate on the distinction between algorithms based on direct gradient measurements and those based on gradient approximations from measurements of the loss function. The prototype gradient-based algorithm is R-M SA, which may be considered a generalization of such techniques as deterministic steepest descent and Newton–Raphson, neural network back-propagation, and infinitesimal perturbation analysis–based optimization for discrete-event systems. The gradient-based algorithms rely on direct measurements of the gradient of the loss function with respect to the parameters being optimized. These measurements typically yield an estimate of the gradient because the underlying data generally include added noise. Because it is not usually the case that one would obtain direct measurements of the gradient (with or without added noise) naturally in the course of operating or simulating a system, one must have detailed knowledge of the underlying system input–output relationships to calculate the R-M gradient estimate from basic system output measurements. In contrast, the approaches based on gradient approximations require only conversion of the basic output measurements to sample values of the loss function, which does not require full knowledge of the system input–output relationships. The classical method for gradient-free stochastic optimization is the Kiefer–Wolfowitz finite-difference SA (FDSA) algorithm.[2]

Because of the fundamentally different information needed in implementing these gradient-based (R-M) and gradient-free algorithms, it is difficult to construct meaningful methods of comparison. As a general rule, however, the gradient-based algorithms will be faster to converge than those using loss function–based gradient approximations when speed is measured in number of iterations. Intuitively, this result is not surprising given the additional information required for the gradient-based algorithms. In particular, on the basis of asymptotic theory, the optimal rate of convergence measured in terms of the deviation of the parameter estimate from the true optimal parameter vector is of order $k^{-1/2}$ for the gradient-based algorithms and of order $k^{-1/3}$ for the algorithms based on gradient approximations, where k represents the number of iterations. (Special cases exist where the maximum rate of convergence for a non-gradient algorithm is arbitrarily close to, or equal to, $k^{-1/2}$.)

In practice, of course, many other factors must be considered in determining which algorithm is best for a given circumstance for the following three reasons: (1) It may not be possible to obtain reliable knowledge of the system input–output relationships, implying that the gradient-based algorithms may be either infeasible (if no system model is available) or undependable (if a poor system model is used). (2) The total cost to achieve effective convergence depends not only on the number of iterations required, but also on the cost needed per iteration, which is typically greater in gradient-based algorithms. (This cost may include greater computational burden, additional human effort required for determining and coding gradients, and experimental costs for model building such as labor, materials, and fuel.) (3) The rates of convergence are based on asymptotic theory and may not be representative of practical convergence rates in finite samples. For these reasons, one cannot say in general that a gradient-based search algorithm is superior to a gradient approximation-based algorithm, even though the gradient-based algorithm has a faster asymptotic rate of convergence (and with simulation-based optimization such as infinitesimal perturbation analysis requires only one system run per iteration, whereas the approximation-based algorithm may require multiple system runs per iteration). As a general rule, however, if direct gradient information is conveniently and reliably available, it is generally to one's advantage to use this information in the optimization process. The focus in this article is the case where such information is not readily available.

The next section describes SPSA and the related FDSA algorithm. Then some of the theory associated with the convergence and efficiency of SPSA is summarized. The following section is an illustration of the implications of the theory in an example related to neural networks. Then practical guidelines for implementation are presented, followed by a summary of some ancillary results and some extensions of the algorithm. Not covered here are global optimization methods such as genetic algorithms and simulated annealing; Spall[3] presents some discussion of such methods in the context of stochastic approximation.

FDSA AND SPSA ALGORITHMS

This article considers the problem of minimizing a (scalar) differentiable loss function $L(\theta)$, where θ is a p-dimensional vector and where the optimization problem can be translated into finding the minimizing θ^* such that $\partial L/\partial\theta = 0$. This is the classical formulation of (local) optimization for differentiable loss functions. It is assumed that measurements of $L(\theta)$ are available at various values of θ. These measurements may or may not include added noise. No direct measurements of $\partial L/\partial\theta$ are assumed available, in contrast to the R-M framework. This section will describe the FDSA and SPSA algorithms. Although the emphasis of this article is SPSA, the FDSA discussion is included for comparison because FDSA is a classical method for stochastic optimization.

The SPSA and FDSA procedures are in the general recursive SA form:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k)\,, \qquad (1)$$

where $\hat{g}_k(\hat{\theta}_k)$ is the estimate of the gradient $g(\theta) \equiv \partial L/\partial\theta$ at the iterate $\hat{\theta}_k$ based on the previously mentioned measurements of the loss function. Under appropriate conditions, the iteration in Eq. 1 will converge to θ^* in some stochastic sense (usually "almost surely") (see, e.g., Fabian[4] or Kushner and Yin[5]).

The essential part of Eq. 1 is the gradient approximation $\hat{g}_k(\hat{\theta}_k)$. We discuss the two forms of interest here. Let $y(\cdot)$ denote a measurement of $L(\cdot)$ at a design level represented by the dot (i.e., $y(\cdot) = L(\cdot)$ + noise) and c_k be some (usually small) positive number. One-sided gradient approximations involve measurements $y(\hat{\theta}_k)$ and $y(\hat{\theta}_k +$ perturbation), whereas two-sided gradient approximations involve measurements of the form $y(\hat{\theta}_k \pm$ perturbation). The two general forms of gradient approximations for use in FDSA and SPSA are finite difference and simultaneous perturbation, respectively, which are discussed in the following paragraphs.

For the finite-difference approximation, each component of $\hat{\theta}_k$ is perturbed one at a time, and corresponding measurements $y(\cdot)$ are obtained. Each component of the gradient estimate is formed by differencing the corresponding $y(\cdot)$ values and then dividing by a difference interval. This is the standard approach to approximating gradient vectors and is motivated directly from the definition of a gradient as a vector of p partial derivatives, each constructed as the limit of the ratio of a change in the function value over a corresponding change in one component of the argument vector. Typically, the ith component of $\hat{g}_k(\hat{\theta}_k)$ ($i = 1, 2,\ldots, p$) for a two-sided finite-difference approximation is given by

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k e_i) - y(\hat{\theta}_k - c_k e_i)}{2c_k}, \qquad (2)$$

where e_i denotes a vector with a one in the ith place and zeros elsewhere (an obvious analogue holds for the one-sided version; likewise for the simultaneous perturbation form below), and c_k denotes a small positive number that usually gets smaller as k gets larger.

The simultaneous perturbation approximation has all elements of $\hat{\theta}_k$ randomly perturbed together to obtain two measurements of $y(\cdot)$, but each component $\hat{g}_{ki}(\hat{\theta}_k)$ is formed from a ratio involving the individual components in the perturbation vector and the difference in the two corresponding measurements. For two-sided simultaneous perturbation, we have

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k \Delta_k) - y(\hat{\theta}_k - c_k \Delta_k)}{2c_k \Delta_{ki}}, \qquad (3)$$

where the distribution of the user-specified p-dimensional random perturbation vector, $\Delta_k = (\Delta_{k1}, \Delta_{k2},\ldots, \Delta_{kp})^T$, satisfies conditions discussed later in this article (superscript T denotes vector transpose).

Note that the number of loss function measurements $y(\cdot)$ needed in each iteration of FDSA grows with p, whereas with SPSA, only two measurements are needed independent of p because the numerator is the same in all p components. This circumstance, of course, provides the *potential* for SPSA to achieve a large savings (over FDSA) in the total number of measurements required to estimate θ when p is large. This potential is realized only if the number of iterations required for effective convergence to θ^* does not increase in a way to cancel the measurement savings per gradient approximation at each iteration. A later section of this article will discuss this efficiency issue further, demonstrating when this potential can be realized by establishing that:

Under reasonably general conditions, SPSA and FDSA achieve the same level of statistical accuracy for a given number of iterations, even though SPSA uses p times fewer function evaluations than FDSA (because each gradient approximation uses only $1/p$ the number of function evaluations).

SELECTED APPLICATIONS OF SPSA

The efficiency issue mentioned in the preceding section (and treated in more detail in the next section) has profound implications for practical multivariate optimization. Many problems that formerly may have been considered intractable with conventional (say, FDSA) methods, may now be solvable. In this section, we summarize three distinct examples, based on work at APL, where SPSA is providing a solution to a problem that appeared intractable using other available methods. In addition, to illustrate some of the other possible applications, we close with a brief summary of some additional projects based on SPSA, most of which have been developed at other institutions.

Signal Timing for Vehicle Traffic Control

A long-standing problem in traffic engineering is to optimize the flow of vehicles through a given road network (Fig. 2). Improving the timing of the traffic signals at intersections in a network is generally the most powerful and cost-effective means of achieving this goal. However, because of the many complex aspects of a traffic system—human behavioral considerations, vehicle flow interactions within the network, weather effects, traffic accidents, long-term (e.g., seasonal) variation, etc.—it has been notoriously difficult to determine the optimal signal timing, especially on a system-wide (multiple intersection) basis. Much of this difficulty has stemmed from the need to build extremely complex models of the traffic dynamics as a component of the control strategy. A "strategy" in this context is a set of rules providing real-time signal timing in response to minute-by-minute changes in the traffic conditions. The APL approach is fundamentally different from those in existence in that it eliminates the need for such complex models. SPSA is central to the approach by providing a means for making small simultaneous changes to all the signal timings in a network and using the information gathered in this way to update the system-wide timing strategy. By avoiding conventional "one-signal-at-a-time" changes to the signal timing strategies, the time it would take to produce an overall optimal strategy for the system is reduced from years or decades (obviously impractical!) to several months (quite reasonable). Note that, unlike the two examples that follow, the savings here is not computational per se, but is inherent in the need for data on a daily basis (and hence represents a reduction in physical experimental costs such as labor and time). This approach is described in detail in Spall and Chin[6] and Chin et al.,[7] including realistic simulations of a 9-intersection network within the central business district of Manhattan, New York, and a 12-intersection network in Montgomery County, Maryland.

Optimal Targeting of Weapon Systems

This is an example of the use of simulations to optimize processes, something done in a wide range of DoD and non-DoD applications. More specifically, given a number of projectiles that are going to be directed at a target, the problem is to optimally select a set of aim points with the goal of maximizing damage to the target while minimizing so-called collateral

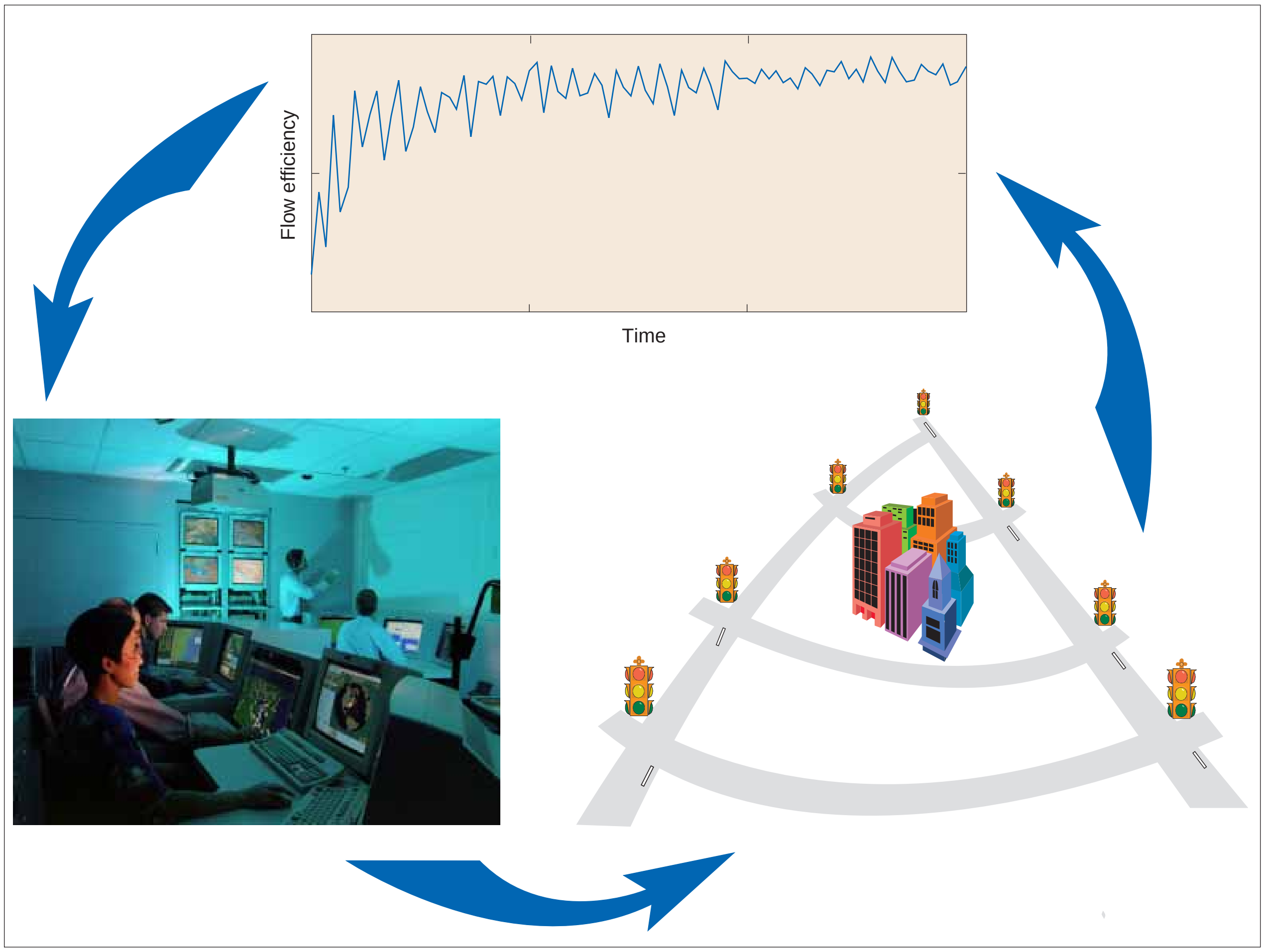

Figure 2. Overall system-wide traffic control concept. Traffic control center provides timing information to signals in traffic network; information on traffic flow is fed back to traffic control center.

damage (damage to sensitive locations not directly associated with the military mission, e.g., schools and hospitals). The projectiles have inherent random variation and may be statistically dependent. So the targeter must allow for the statistical variation between the aim point and actual impact point in developing a strategy for determining aim points. In such cases it is desirable to use patterning of multiple projectiles. "Patterning" in this context means aiming the projectiles at a set of points that may not overlap each other or be within the target boundaries. Figure 3 illustrates the concept for one target and five projectiles; a bias away from the "stay-out zone" (e.g., a civilian facility) is apparent in this case where it is desired to destroy the target while minimizing the chances of producing damage to the stay-out zone. For scenarios with many projectiles that are independently targeted, the damage function—which must be evaluated to find the best aim point pattern—is likely to be analytically unavailable and will require estimation by Monte Carlo simulation. In particular, to evaluate the effectiveness of a given set of aim points, it is necessary to run one or more simulations (recall that there is random variation in the outcome of one "volley" of projectiles corresponding to one simulation). Commonly used techniques for solving the optimization problem by simulation are computationally intensive and prohibitively time-consuming since the damage function for many different sets of aim points must be evaluated (i.e., many simulations must be run). The SPSA method provides an efficient means of solving this multivariate problem, which for planar targets has a dimension of $p = 2 \times$ [no. of projectiles] (so $p = 10$ in the small-scale example of Fig. 3). SPSA works by varying all of the aim point coordinates simultaneously and running a simulation in the process of producing the gradient approximation for the optimization process. This procedure is repeated as the iteration for the optimization proceeds. This method contrasts significantly with conventional methods where one would vary only one of the coordinates for one of the aim points prior to running a simulation, repeating that process as each coordinate for each aim point was changed at a specified nominal set of aim points to construct a gradient approximation at the

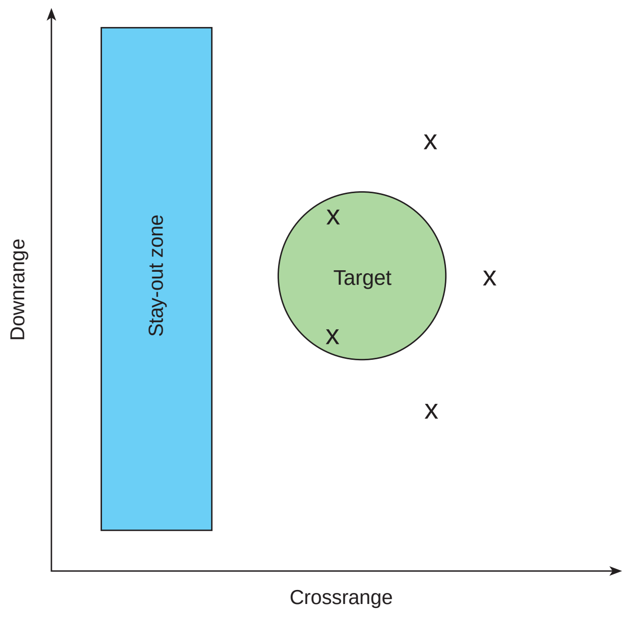

Figure 3. Example of optimal aim points (×) given stay-out zone.

given nominal point. The process is repeated as the nominal aim points are varied over the course of the optimization. By simultaneously changing the aim points, one is able to reduce by a factor of p the number of simulations needed, possibly reducing the run times from days to minutes or hours. A more complete description of this approach to determining aim points is given in Heydon et al.[8] and Hill and Heydon.[9]

Locating Buried Objects via Electrical Conductivity

ECOL (electrical conductivity object locator) is an approach to determining the location of buried objects via injecting electrical current into the ground in an area surrounding a candidate object. Measurements of the electric potential are taken near the surface, which then form the basis for constructing a subsurface characterization of the conductivity. The object being sought must have conductivity different from the surrounding soil, which would include metal or plastic objects of potential interest in mine sweeping or buried waste detection. Present technology is limited to searching for objects from 5 to 500 cm below the surface in an area ranging from 10 to 30 m^2. Several field demonstrations have been conducted on APL property, and the setup for one is shown in Fig. 4. The basis for ECOL is to use the contrast in conductivity between the buried object and surrounding soil to construct a finite-element model of the subsurface. This represents a demanding optimization problem attributable to the uncertainty about the nature of the subsurface and the many potential impurities (stones, sticks, tree roots,

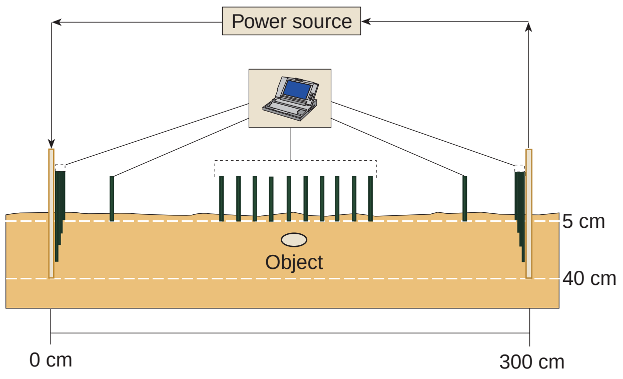

Figure 4. APL demonstration site for ECOL. Indicated probes measure the electric potential.

etc.) that can affect conductivity and the high dimensionality of the finite-element model. The inherent uncertainty about the subsurface makes gradient-based methods infeasible, and the high dimensionality makes the “one-variable-at-a-time” methods very time-consuming. SPSA was used to provide a relatively easy and rapid solution to this problem. For example, with surface data from one of the field experiments, effective convergence of the algorithm was achieved after about 4 min on a 180-MHz Pentium PC; the conventional finite-difference method would have taken approximately 6 to 7 h on the same PC. Larger-scale practical implementations would show an even much larger relative savings. A description of ECOL and some of the field experiments is given in Chin and Srinivasan.[10]

Some Other Applications

Some additional recent applications of SPSA, initiated both in and out of APL, are described in Hill and Fu[11] and Fu and Hill[12] (queuing systems), Hopkins[13] (control of a heavy ion beam), Rezayat[14] (industrial quality improvement), Maeda et al.[15] (pattern recognition), Kleinman et al.[16] (simulation-based optimization with applications to air traffic management), Cauwenberghs[17] (neural network training), Spall and Cristion[18,19] and Maeda and De Figueiredo[20] (neural network training for adaptive control of dynamic systems), Gerencsér[21] (classification of ECG signals for heart monitoring), Luman[22] (simulation-based decision aiding), Alessandri and Parisini[23] (statistical model parameter estimation/fault detection), Nechyba and Xu[24] (human–machine interaction), Sadegh and Spall[25] (sensor placement and configuration), and Chin[26] (signal inversion for a complex physical model).

BASIC ASSUMPTIONS AND SUPPORTING THEORY

With the goal of minimizing a loss function $L(\theta)$ over feasible values of θ, the SPSA algorithm works by iterating from an initial guess of the optimal θ, where the iteration process depends on the aforementioned simultaneous perturbation approximation to the gradient $g(\theta)$.

Spall[27,28] presents sufficient conditions for convergence of the SPSA iterate ($\hat{\theta}_k \rightarrow \theta^*$ in the stochastic "almost sure" sense) using a differential equation approach well known in general SA theory (e.g, Kushner and Yin[5]). To establish convergence, conditions are imposed on both gain sequences (a_k and c_k), the user-specified distribution of Δ_k, and the statistical relationship of Δ_k to the measurements $y(\cdot)$. We will not repeat the conditions here since they are available in Spall.[27,28] The essence of the main conditions is that a_k and c_k both go to 0 at rates neither too fast nor too slow, that $L(\theta)$ is sufficiently smooth (several times differentiable) near θ^*, and that the $\{\Delta_{ki}\}$ are independent and symmetrically distributed about 0 with finite inverse moments $E(|\Delta_{ki}|^{-1})$ for all k, i. One particular distribution for Δ_{ki} that satisfies these latter conditions is the symmetric Bernoulli ± 1 distribution; two common distributions that do *not* satisfy the conditions (in particular, the critical finite inverse moment condition) are the uniform and normal distributions.

In addition to establishing the formal convergence of SPSA, Spall (Ref. 28, Sect. 4) shows that the probability distribution of an appropriately scaled $\hat{\theta}_k$ is approximately normal (with a specified mean and covariance matrix) for large k. This asymptotic normality result, together with a parallel result for FDSA, can be used to study the relative efficiency of SPSA. This efficiency is the major theoretical result justifying the use of SPSA. The efficiency depends on the shape of $L(\theta)$, the values for $\{a_k\}$ and $\{c_k\}$, and the distributions of the $\{\Delta_{ki}\}$ and measurement noise terms. There is no single expression that can be used to characterize the relative efficiency; however, as discussed in Spall (Ref. 28, Sect. 4) and Chin,[29] in most practical problems SPSA will be asymptotically more efficient than FDSA. For example, if a_k and c_k are chosen as in the guidelines of Spall,[30] then by equating the asymptotic mean-squared error $E(\|\hat{\theta}_k - \theta^*\|^2)$ in SPSA and FDSA, we find

$$\frac{\text{No. of measurements of } L(\theta) \text{ in SPSA}}{\text{No. of measurements of } L(\theta) \text{ in FDSA}} \rightarrow \frac{1}{p} \qquad (4)$$

as the number of loss measurements in both procedures gets large. Hence, Expression 4 implies that the p-fold savings per iteration (gradient approximation) translates directly into a p-fold savings in the overall optimization process despite the complex nonlinear ways in which the sequence of gradient approximations manifests itself in the ultimate solution $\hat{\theta}_k$.

Relative to implementation in a practical problem, another way of looking at Expression 4 is that:

One properly chosen simultaneous random change in all the variables in a problem provides as much information ***for optimization*** as a full set of one-at-a-time changes of each variable.

This surprising and significant result seems to run counter to all that one learns in engineering and scientific training. It is the qualifier "for optimization" that is critical to the validity of the statement. To view an online animated demonstration of this concept, select the blue box.

Let us provide some informal mathematical rationale for this key result. Figure 5 provides an example of a two-variable problem, where the level curves show points of equal value in the loss function. In a low- or no-noise setting, the FDSA algorithm will behave similarly to a traditional gradient descent algorithm in taking steps that provide the locally greatest reduction in the loss function. A standard result in calculus shows that this "steepest descent" direction is perpendicular to the level curve at that point, as shown in the steps for the FDSA algorithm of Fig. 5 (each straight segment is perpendicular to the level curve at the origin of the segment). Hence, the FDSA algorithm is behaving much as an aggressive skier might act in descending a hill by going in small segments that provide the steepest drop from the start of each segment. SPSA, on the other hand, with its random search direction, does not follow the path of locally steepest descent. On average, though, it will nearly follow the steepest descent path because the gradient approximation is an almost unbiased estimator of the gradient (i.e., $E[\hat{g}_k(\theta)] = g(\theta)$ + small bias, where small bias is proportional to c_k^2, and c_k is the small number mentioned earlier). Over the course of many iterations, the errors associated with the "misdirections" in SPSA will average out in a manner analogous to the way random errors cancel out in forming the sample mean of almost any random process (the a_k sequence in Eq. 1 governs this averaging). Figure 5 shows this effect at work in the way the SPSA search direction tends to "bounce around" the FDSA search direction, while ultimately settling down near the solution in the same number of steps. Although this discussion was motivated by the two-variable ($p = 2$) problem with no- or low-noise loss function measurements (so that the FDSA algorithm behaves very nearly like a true gradient descent algorithm), the same essential intuition applies in higher-dimension settings and noisier loss measurements. Noisy loss measurements imply that the FDSA algorithm will also not closely track a gradient descent algorithm as in Fig. 5; however, the relationship between SPSA and FDSA (which is what Expression 4 pertains to) will still be governed by the idea of averaging out the errors in directions over a large number of iterations.

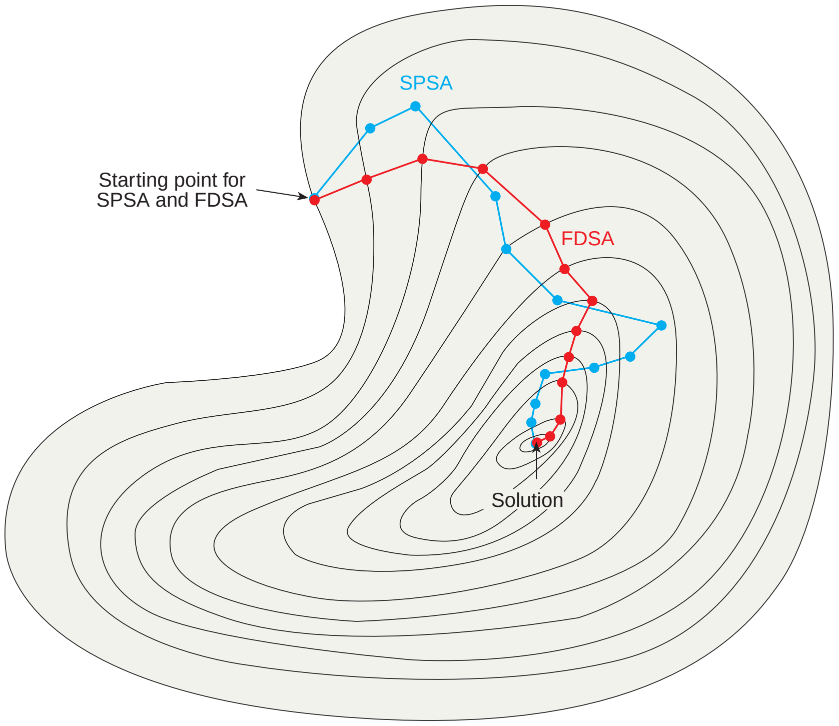

Figure 5. Example of relative search paths for SPSA and FDSA in $p = 2$ problem. Deviations of SPSA from FDSA average out in reaching a solution in the same number of iterations; FDSA nearly follows the gradient descent path (perpendicular to level curves) in the low-noise setting.

EXPERIMENTAL RESULTS

Figure 6 shows the implications of the theoretical result just discussed in a practical setting. This graph shows the results of a simulation using a neural network to regulate the water purity and methane gas by-product of a wastewater treatment process (Spall and Cristion[19] discuss this problem in detail). The vertical axis represents a normalized (for units) version of the loss function $L(\theta)$ (measuring the deviation from certain target values in water cleanliness and methane gas by-product), and the horizontal axis measures the iterations of the algorithms. The dimension of the θ vector in this case was 412, corresponding to the number of "connection weights" in the neural network. The essential point to observe in this graph is that the FDSA and SPSA approaches achieve very similar levels of accuracy for a given number of iterations after the first few iterations, but that SPSA only uses 2 experiments (loss evaluations) per iteration, whereas FDSA uses 824 experiments! This difference obviously leads to a very substantial savings—representing computational savings in a simulation exercise or direct time and money savings in a real waste treatment plant—in the overall problem of estimating the neural network weights. (The sharp initial decline of FDSA in Fig. 6 may be slightly misleading because the weights had yet to begin stabilizing and because the number of measurements is still large: over 1600 in the first two FDSA iterations versus 160 for all the iterations of SPSA.)

The numerical study in Fig. 6 is only one of many such examples. Further, there have been comparisons with other types of stochastic optimization algorithms. For example, Chin[26] performs a comparison with the popular simulated annealing algorithm in the context of model estimation for a magnetosphere model and finds that SPSA significantly outperforms simulated annealing. The author has also performed comparisons with simulated annealing and several types of directed random search, and found similar superior relative performance. An open issue is to conduct a careful comparison with the popular genetic algorithm and related evolutionary methods.

IMPLEMENTATION OF SPSA

The following step-by-step summary shows how SPSA iteratively produces a sequence of estimates. The boxed insert presents an implementation of the following steps in MATLAB code.

Step 1: *Initialization and coefficient selection*. Set counter index $k = 1$. Pick initial guess and non-negative coefficients a, c, A, α, and γ in the SPSA gain sequences $a_k = a/(A + k)^\alpha$ and $c_k = c/k^\gamma$. The choice of the gain sequences (a_k and c_k) is critical to the performance of SPSA (as with all stochastic optimization algorithms

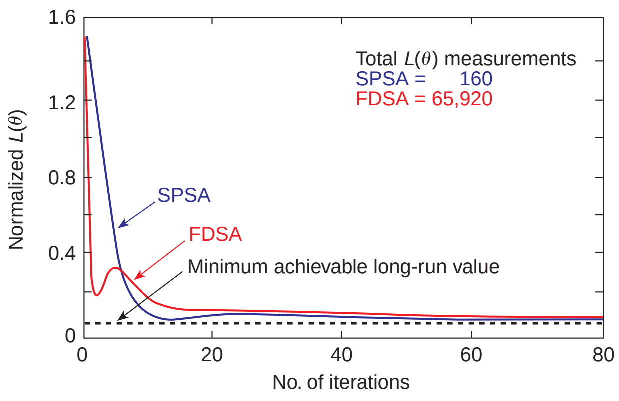

Figure 6. Relative performance of SPSA and FDSA for controller in a wastewater treatment system.

MATLAB CODE

The accompanying figure presents a sample MATLAB code for performing n iterations of the standard (first-order) SPSA algorithm. Algorithm initialization for program variables `theta, n, p, a, A, c, alpha, gamma` is not shown here since that can be handled in many ways (e.g., read from another file, direct inclusion in the program, and user input during execution). The program calls an external function "loss" to obtain the (possibly noisy) measurements. The Δ_{ki} elements are generated according to a Bernoulli ±1 distribution.

```
For  k=1:n
  ak=a/(k+A)^alpha;
  ck=c/k^gamma;
  delta=2*round(rand(p,1))-1;
  thetaplus=theta+ck*delta;
  thetaminus=theta-ck*delta;
  yplus=loss(thetaplus);
  yminus=loss(thetaminus);
  ghat=(yplus-yminus)./(2*ck*delta);
  theta=theta-ak*ghat;
end
theta
```

If maximum and minimum values on the values of `theta` can be specified, say `thetamax` and `thetamin`, then the following two lines can be added below the `theta` update line to impose the constraints

```
  theta=min(theta, thetamax);
  theta=max(theta, thetamin);
```

and the choice of their respective algorithm coefficients). Spall[30] provides some guidance on picking these coefficients in a practically effective manner. (In cases where the elements of θ have very different magnitudes, it may be desirable to use a matrix scaling of the gain a_k if prior information is available on the relative magnitudes. The next section discusses a second-order version of SPSA that automatically scales for different magnitudes.)

Step 2: *Generation of the simultaneous perturbation vector.* Generate by Monte Carlo a p-dimensional random perturbation vector Δ_k, where each of the p components of Δ_k are independently generated from a zero-mean probability distribution satisfying the preceding conditions. A simple (and theoretically valid) choice for each component of Δ_k is to use a Bernoulli ±1 distribution with probability of 1/2 for each ±1 outcome. Note that uniform and normal random variables are not allowed for the elements of Δ_k by the SPSA regularity conditions (since they have infinite inverse moments).

Step 3: *Loss function evaluations.* Obtain two measurements of the loss function $L(\cdot)$ based on the simultaneous perturbation around the current $\hat{\theta}_k$:$y(\hat{\theta}_k + c_k\Delta_k)$ and $y(\hat{\theta}_k - c_k\Delta_k)$ with the c_k and Δ_k from Steps 1 and 2.

Step 4: *Gradient approximation.* Generate the simultaneous perturbation approximation to the unknown gradient $g(\hat{\theta}_k)$:

$$\hat{g}_k(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k - c_k\Delta_k)}{2c_k}\begin{bmatrix}\Delta_{k1}^{-1}\\ \Delta_{k2}^{-1}\\ \vdots \\ \Delta_{kp}^{-1}\end{bmatrix}, \qquad (5)$$

where Δ_{ki} is the ith component of the Δ_k vector (which may be ±1 random variables as discussed in Step 2); note that the common numerator in all p components of $\hat{g}_k(\hat{\theta}_k)$ reflects the simultaneous perturbation of all components in $\hat{\theta}_k$ in contrast to the component-by-component perturbations in the standard finite-difference approximation.

Step 5: *Updating θ estimate.* Use the standard SA form

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k\hat{g}_k(\hat{\theta}_k) \qquad (6)$$

to update $\hat{\theta}_k$ to a new value $\hat{\theta}_{k+1}$. Modifications to the basic updating step in Eq. 6 are sometimes desirable to enhance convergence and impose constraints. These modifications block or alter the update to the new value of θ if the "basic" value from Eq. 6 appears undesirable. Reference 31 (Sect. 2) discusses several possibilities. One easy possibility if maximum and minimum allowable values can be specified on the components of θ is shown at the bottom of the boxed insert.

Step 6: *Iteration or termination.* Return to Step 2 with k+1 replacing k. Terminate the algorithm if there is little change in several successive iterates or the maximum allowable number of iterations has been reached (more formal termination guidance is discussed in Pflug, Ref. 32, pp. 297–300).

FURTHER RESULTS AND EXTENSIONS TO BASIC SPSA ALGORITHM

Sadegh and Spall[33] consider the problem of choosing the best distribution for the Δ_k vector. On the basis of asymptotic distribution results, it is shown that the optimal distribution for the components of Δ_k is symmetric Bernoulli. This simple distribution has also proven effective in many finite-sample practical and simulation examples. The recommendation in Step 2 of the algorithm description follows from these findings. (It should be noted, however, that other distributions are sometimes desirable. Since the user has full control over this choice and since the generation of Δ_k represents a trivial cost toward the optimization, it may be worth evaluating other possibilities in some applications. For example, Maeda and De Figueiredo[20] used a symmetric two-part uniform distribution, i.e., a uniform distribution with a section removed near 0 [to preserve the finiteness of inverse moments], in an application for robot control.)

Some extensions to the basic SPSA algorithm are reported in the literature. For example, its use in feedback control problems, where the loss function changes with time, is given in Spall and Cristion.[18,19,34] Reference 34 is the most complete methodological and theoretical treatment. Reference 18 also reports on a gradient smoothing idea (analogous to "momentum" in the neural network literature) that may help reduce noise effects and enhance convergence (and also gives guidelines for how the smoothing should be reduced over time to ensure convergence). Alternatively, it is possible to average several simultaneous perturbation gradient approximations at each iteration to reduce noise effects (at the cost of additional function measurements); this is discussed in Spall.[28] An implementation of SPSA for global minimization is discussed in Chin[35] (i.e., the case where there are multiple minimums at which $g(\theta) = 0$); this approach is based on a step-wise (slowly decaying) sequence c_k (and possibly a_k). The problem of constrained (equality and inequality) optimization with SPSA is considered in Sadegh[36] and Fu and Hill[12] using a projection approach. A one-measurement form of the simultaneous perturbation gradient approximation is considered in Spall[37]; although it is shown in Ref. 37 that the standard two-measurement form will usually be more efficient (in terms of total number of loss function measurements to obtain a given level of accuracy in the θ iterate), there are advantages to the one-measurement form in real-time operations where the underlying system dynamics may change too rapidly to get a credible gradient estimate with two successive measurements.

An "accelerated" form of SPSA is reported in Spall.[31,38] This approach extends the SPSA algorithm to include second-order (Hessian) effects with the aim of accelerating convergence in a stochastic analogue to the deterministic Newton–Raphson algorithm. Like the standard (first-order) SPSA algorithm, this second-order algorithm is simple to implement and requires only a small number—independent of p—of loss function measurements per iteration (no gradient measurements, as in standard SPSA). In particular, only four measurements are required to estimate the loss-function gradient and inverse Hessian at each iteration (and one additional measurement is sometimes recommended as a check on algorithm behavior). The algorithm is implemented with two simple parallel recursions: one for θ and one for the Hessian matrix of $L(\theta)$. The recursion for θ is a stochastic analogue of the well-known Newton–Raphson algorithm of deterministic optimization. The recursion for the Hessian matrix is simply a recursive calculation of the sample mean of per-iteration Hessian estimates formed using SP-type ideas.

CONCLUSION

Relative to standard deterministic methods, stochastic optimization considerably broadens the range of practical problems for which one can find rigorous optimal solutions. Algorithms of the stochastic optimization type allow for the effective treatment of problems in areas such as network analysis, simulation-based optimization, pattern recognition and classification, neural network training, image processing, and nonlinear control. It is expected that the role of stochastic optimization will continue to grow as modern systems increase in complexity and as population growth and dwindling natural resources force trade-offs that were previously unnecessary.

The SPSA algorithm has proven to be an effective stochastic optimization method. Its primary virtues are ease of implementation and lack of need for loss function gradient, theoretical and experimental support for relative efficiency, robustness to noise in the loss measurements, and empirical evidence of ability to find a global minimum when multiple (local and global) minima exist. SPSA is primarily limited to continuous-variable problems and, relative to other methods, is most effective when the loss function measurements include added noise. Numerical comparisons with techniques such as the finite-difference method, simulated annealing, genetic algorithms, and random search have supported the claims of SPSA's effectiveness in a wide range of practical problems. The rapidly growing number of applications throughout the world provides further evidence of the algorithm's effectiveness. To add to the effectiveness, there have been some extensions of the basic idea, including a stochastic analogue of the fast deterministic Newton–Raphson (second-order) algorithm, adaptations for real-time (control) implementations, and versions for some types of constrained and global optimization problems. Although much work continues in extending the basic algorithm to a broader range of real-world settings, SPSA addresses a wide range of difficult problems and should likely be considered for many of the stochastic optimization challenges encountered in practice.

REFERENCES

[1]Robbins, H., and Monro, S., "A Stochastic Approximation Method," *Ann. Math. Stat.* 29, 400–407 (1951).
[2]Kiefer, J., and Wolfowitz, J., "Stochastic Estimation of a Regression Function," *Ann. Math. Stat.* 23, 462–466 (1952).
[3]Spall, J. C., "Stochastic Optimization, Stochastic Approximation, and Simulated Annealing," in *Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster (ed.), Wiley, New York (in press, 1999).

[4] Fabian, V., "Stochastic Approximation," in *Optimizing Methods in Statistics,* J. J. Rustigi (ed.), Academic Press, New York, pp. 439–470 (1971).

[5] Kushner, H. J., and Yin, G. G., *Stochastic Approximation Algorithms and Applications,* Springer-Verlag, New York (1997).

[6] Spall, J. C., and Chin, D. C., "Traffic-Responsive Signal Timing for System-Wide Traffic Control," *Transp. Res., Part C* 5, 153–163 (1997).

[7] Chin, D. C., Spall, J. C., and Smith, R. H., *Evaluation and Practical Considerations for the S-TRAC System-Wide Traffic Signal Controller*, Transportation Research Board 77th Annual Meeting, Preprint 98-1230 (1998).

[8] Heydon, B. D., Hill, S. D., and Havermans, C. C., "Maximizing Target Damage Through Optimal Aim Point Patterning" in *Proc. AIAA Conf. on Missile Sciences*, Monterey, CA (1998).

[9] Hill, S. D., and Heydon, B. D., *Optimal Aim Point Patterning*, Report SSD/PM-97-0448, JHU/APL, Laurel, MD (25 Jul 1997).

[10] Chin, D. C., and Srinivasan, R., "Electrical Conductivity Object Locator," in *Proc. Forum '97—A Global Conf. on Unexploded Ordnance*, Nashville, TN, pp. 50–57 (1998).

[11] Hill, S. D., and Fu, M. C., "Transfer Optimization via Simultaneous Perturbation Stochastic Approximation," in *Proc. Winter Simulation Conf.,* C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, D. (eds.), pp. 242–249 (1995).

[12] Fu, M. C., and Hill, S. D., "Optimization of Discrete Event Systems via Simultaneous Perturbation Stochastic Approximation," *Trans. Inst. Industr. Eng.* 29, 233–243 (1997).

[13] Hopkins, H. S., *Experimental Measurement of a 4-D Phase Space Map of a Heavy Ion Beam*, Ph.D. thesis, Dept. of Nuclear Engineering, University of California—Berkeley (Dec 1997).

[14] Rezayat, F., "On the Use of an SPSA-Based Model-Free Controller in Quality Improvement," *Automatica* 31, 913–915 (1995).

[15] Maeda, Y., Hirano, H., and Kanata, Y., "A Learning Rule of Neural Networks via Simultaneous Perturbation and Its Hardware Implementation," *Neural Networks* 8, 251–259 (1995).

[16] Kleinman, N. L., Hill, S. D., and Ilenda, V. A., "SPSA/SIMMOD Optimization of Air Traffic Delay Cost," in *Proc. American Control Conf.*, pp. 1121–1125 (1997).

[17] Cauwenberghs, G., *Analog VLSI Autonomous Systems for Learning and Optimization*, Ph.D. thesis, California Institute of Technology (1994).

[18] Spall, J. C., and Cristion, J. A., "Nonlinear Adaptive Control Using Neural Networks: Estimation Based on a Smoothed Form of Simultaneous Perturbation Gradient Approximation," *Stat. Sinica* 4, 1–27 (1994).

[19] Spall, J. C., and Cristion, J. A., "A Neural Network Controller for Systems with Unmodeled Dynamics with Applications to Wastewater Treatment," *IEEE Trans. Syst., Man, Cybernetics–B* 27, 369–375 (1997).

[20] Maeda, Y., and De Figueiredo, R. J. P., "Learning Rules for Neuro-Controller via Simultaneous Perturbation," *IEEE Trans. Neural Networks* 8, 1119–1130 (1997).

[21] Gerencsér, L., "The Use of the SPSA Method in ECG Analysis," *IEEE Trans. Biomed. Eng.* (in press, 1998).

[22] Luman, R. R., *Quantitative Decision Support for Upgrading Complex Systems of Systems*, Ph.D. thesis, School of Engineering and Applied Science, George Washington University (1997).

[23] Alessandri, A., and Parisini, T., "Nonlinear Modelling of Complex Large-Scale Plants Using Neural Networks and Stochastic Approximation," *IEEE Trans. Syst., Man, Cybernetics—A* 27, 750–757 (1997).

[24] Nechyba, M. C., and Xu, Y., "Human-Control Strategy: Abstraction, Verification, and Replication," *IEEE Control Syst. Magazine* 17(5), 48–61 (1997).

[25] Sadegh, P., and Spall, J. C., "Optimal Sensor Configuration for Complex Systems," in *Proc. Amer. Control Conf.,* Philadelphia, PA, 3575–3579 (1998).

[26] Chin, D. C., "The Simultaneous Perturbation Method for Processing Magnetospheric Images," *Opt. Eng.* (in press, 1999).

[27] Spall, J. C., "A Stochastic Approximation Algorithm for Large-Dimensional Systems in the Kiefer-Wolfowitz Setting," in *Proc. IEEE Conf. on Decision and Control*, pp. 1544–1548 (1988).

[28] Spall, J. C., "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Trans. Autom. Control* 37, 332–341 (1992).

[29] Chin, D. C., "Comparative Study of Stochastic Algorithms for System Optimization Based on Gradient Approximations," *IEEE Trans. Syst., Man, and Cybernetics—B* 27, 244–249 (1997).

[30] Spall, J. C., "Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization," *IEEE Trans. Aerosp. Electron. Syst.* 34(3), 817–823 (1998).

[31] Spall, J. C., *Adaptive Simultaneous Perturbation Method for Accelerated Optimization,* Memo PSA-98-017, JHU/APL, Laurel, MD (1998).

[32] Pflug, G. Ch., *Optimization of Stochastic Models: The Interface Between Simulation and Optimization*, Kluwer Academic, Boston (1996).

[33] Sadegh, P., and Spall, J. C., "Optimal Random Perturbations for Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," in *Proc. American Control Conf.,* pp. 3582–3586 (1997).

[34] Spall, J. C., and Cristion, J. A., "Model-Free Control of Nonlinear Stochastic Systems with Discrete-Time Measurements," *IEEE Trans. Autom. Control* 43, 1198–1210 (1998).

[35] Chin, D. C., "A More Efficient Global Optimization Algorithm Based on Styblinski and Tang," *Neural Networks* 7, 573–574 (1994).

[36] Sadegh, P., "Constrained Optimization via Stochastic Approximation with a Simultaneous Perturbation Gradient Approximation," *Automatica* 33, 889–892 (1997).

[37] Spall, J. C., "A One-Measurement Form of Simultaneous Perturbation Stochastic Approximation," *Automatica* 33, 109–112 (1997).

[38] Spall, J. C., "Accelerated Second-Order Stochastic Optimization Using Only Function Measurements," in *Proc. 36th IEEE Conf. on Decision and Control,* pp. 1417–1424 (1997).

ACKNOWLEDGMENTS: This work was partially supported by U.S. Navy contract N00024-98-D-8124 and the APL Independent Research and Development (IR&D) Program. Parts of this article have benefited from helpful comments of Gert Cauwenberghs, Vaclav Fabian, Michael Fu, John L. Maryak, Boris T. Polyak, M. A. Styblinski, and Sid Yakowitz. I would also like to thank Larry Levy, the Strategic Systems Department, and the IR&D Committee for providing an environment in which initially "high risk" concepts like this algorithm can be developed into broadly useful methods.

THE AUTHOR

JAMES C. SPALL joined APL's Strategic Systems Department in 1983 after receiving an S.M. from M.I.T. and a Ph.D. from the University of Virginia. He was appointed to the Principal Professional Staff in 1991. He also teaches in the JHU Whiting School of Engineering. Dr. Spall has published many articles in the areas of statistics and control and holds two U.S. patents. In 1990, he received the Hart Prize as principal investigator of an outstanding Independent Research and Development project at APL, and in 1997 he was the Chairman of the 4th Symposium on Research and Development at APL. He is an Associate Editor for the *IEEE Transactions on Automatic Control* and a Contributing Editor for the *Current Index to Statistics*. He also was the editor and co-author for the book *Bayesian Analysis of Time Series and Dynamic Models* (Marcel Dekker), and is the author of the forthcoming book *Introduction to Stochastic Search and Optimization* (Wiley). Dr. Spall is a senior member of IEEE, a member of the American Statistical Association, and a fellow of the engineering honor society Tau Beta Pi. His e-mail address is james.spall@jhuapl.edu.