Global Random Optimization by Simultaneous Perturbation Stochastic Approximation

John L. Maryak and Daniel C. Chin

Practitioners of iterative optimization techniques want their chosen algorithm to reach the global optimum rather than get stranded at a local optimum value. In this article, we discuss two theorems on the global convergence of an algorithm called Simultaneous Perturbation Stochastic Approximation (SPSA) that has performed well in complex optimization problems. The first provides conditions under which SPSA will converge in probability to a global optimum using the well-known method of injected noise. In the second theorem we show that, under different conditions, "basic" SPSA *without injected noise* can achieve convergence in probability to a global optimum. This global convergence without injected noise can have important benefits in the setup and performance of the algorithm. We present a numerical study comparing the global convergence of SPSA to that of a genetic algorithm.

INTRODUCTION

Simultaneous Perturbation Stochastic Approximation (SPSA) is a powerful algorithm for optimization in complex systems. This APL-developed algorithm has been successfully used in numerous applications around the world. This article describes two theoretical results pertaining to the global convergence performance of SPSA. We set the stage in this section with a discussion of optimization, stochastic optimization, and stochastic approximation (SA) in general. We then briefly describe the SPSA algorithm, introduce the issue of global convergence of a recursive optimization scheme, and present our main results.

Optimization Algorithms

Many, if not most, problems in mathematical optimization can be expressed as finding the setting of certain "adjustable" parameters so as to minimize a "loss" function. To make this discussion more concrete, consider the problem of setting traffic light timing schedules (the length of the red, yellow, and green cycles) in a portion of a city to minimize the total time spent by vehicles in that area waiting at intersections during rush hour (Fig. 1). In this context, the parameters are all of the numbers needed to express the traffic light timing strategy, and the loss function is the resulting average (over several days, say) total delay time for vehicles in that area during rush hour. In the following discussion, we suppose that the parameters are strung together into a column of numbers, a vector that we denote by θ. The loss that occurs using the setting θ is denoted by $L(\theta)$. Typically, the loss function is a single number (scalar) expressing a quantity, like the average total delay time, that one wants to minimize. Suppose for now that there

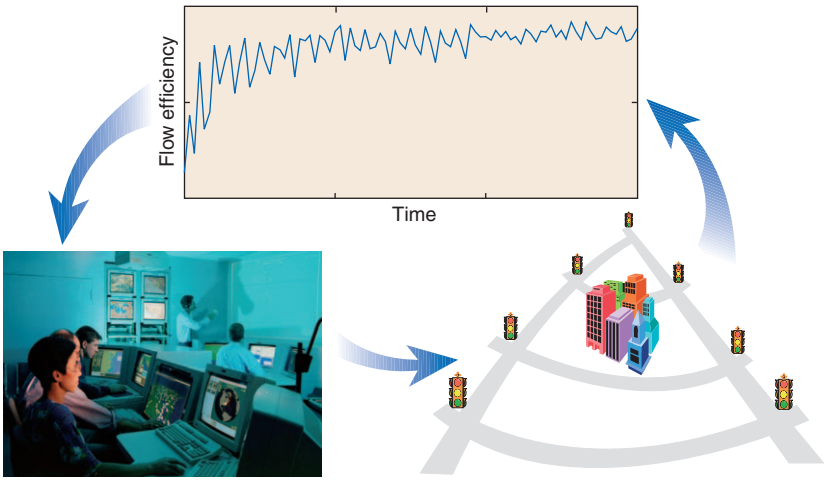

Figure 1. Overall system-wide traffic control concept. The traffic control center provides timing information to signals in the traffic network, and information on traffic flow is fed back to the traffic control center.

is a unique single value of θ that minimizes $L(\theta)$. We will designate this optimum value of θ as θ^*.

Optimization algorithms are ubiquitous and essential in our technical society, with applications ranging from control systems, estimation, modeling, design methods, and system simulation to stock market analysis. At APL, optimization methods have been used in optimal targeting of weapon systems, signal timing for traffic control, locating buried ordnance, and optimization of queuing and missile guidance systems.

Many approaches have been devised for numerous applications over the long history of this type of problem. Perhaps the most common approach is to use an iterative (recursive) search algorithm, i.e., an algorithm that starts with an initial guess, say $\hat{\theta}_0$, of the value of θ^*, and then uses some information about the loss function at the current guess $\hat{\theta}_k$, to compute the next guess, $\hat{\theta}_{k+1}$, as to the value of θ^*. These steps are repeated to improve the guesses as the iterations continue so that eventually the iterates $\hat{\theta}_k$ converge to θ^*. Figure 2 is a simple example of this process where the dimension of θ is 2. In these algorithms, the "information about the loss function" often includes some form of information about the gradient (derivative), $\partial L(\hat{\theta}_k)/\partial\theta$, of the loss function at $\hat{\theta}_k$. The gradient is a vector that expresses information about the slope of the loss function at the current iterate $(\hat{\theta}_k)$ of θ, which can be useful in indicating where to look for a better value of θ. In the discussion here, we will assume that these derivatives are well defined (i.e., exist mathematically) and refer to the gradient as $g(\theta) \equiv \partial L(\theta)/\partial\theta$.

Stochastic Search Algorithms and Stochastic Approximation

The term *stochastic optimization* is used to indicate that there is some randomness in the optimization problem scenario. A common example is an application in which only noisy measurements of the loss function are available. For example, in the traffic control problem mentioned earlier, it is clear that any particular average of measured delay times will only be an approximation to the true long-term average value, i.e., the computed average will be a noisy measurement of $L(\theta)$. To reflect this condition, let us define the measurement actually obtained as $y(\theta) = L(\theta) + \textit{measurement noise}$, where the measurement noise may or may not be zero. A common approach to handling stochastic optimizations is to use a search algorithm that includes some form of averaging in hopes of "averaging out" the effects of the randomness. A useful software "toolbox" for stochastic optimization has been developed at APL by Brian Reardon[1] in connection with work on the performance optimization of missile guidance and control algorithms.[2]

We want to focus here on one very popular type of search algorithm called *stochastic approximation*. The general recursive SA form is

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

where $\{a_k\}$ is a sequence of positive scalar coefficients (often called "gains") that typically decreases to zero, and $\hat{g}_k(\hat{\theta}_k)$ is either the gradient at $\hat{\theta}_k$ or an approximation (possibly corrupted by noise) to the gradient $g(\hat{\theta}_k)$ computed at the kth step of the algorithm. Here, $k = 1, 2, 3, \ldots$ counts the iterations of the algorithm, which essentially repeats Eq. 1 over and over until (it is hoped)

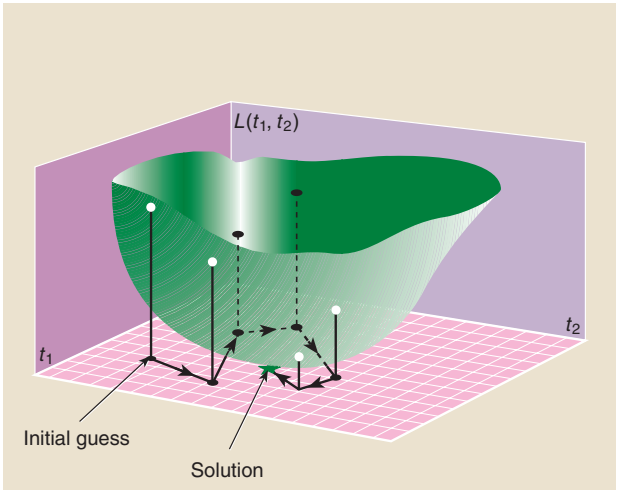

Figure 2. Example of a stochastic optimization algorithm minimizing the loss function $L(t_1, t_2)$. Height of vertical line after each optimization step denotes loss function value.

the iterates $\hat{\theta}_k$ converge to θ^*. The general form shown in Eq. 1 occurs in many well-known algorithms such as neural network backpropagation, steepest descent, least-mean-squares, and simulation-based optimization. Intuitively, the minus sign in the equation indicates that the algorithm is moving, at least roughly, in the "gradient descent" direction, i.e., downhill along the loss function surface, toward the minimum. Also, the recursive form of the algorithm effectively results in a weighted averaging of the gradient terms, which, under the proper conditions, provides a useful averaging effect on the inherent randomness. SA algorithms have been used effectively for many years, and a large body of theory exists describing the convergence and other properties of these algorithms (see, e.g., Kushner and Yin[3]).

Approximate-Gradient Stochastic Approximation

In the field of SA, a distinction is often made between problems in which a direct, possibly noisy, measurement or computation of the gradient is available and those in which it is not. The available-gradient type of SA is often referred to as the Robbins-Monro SA, and the approximate-gradient type (also known as the gradient-free type) is usually called the Kiefer-Wolfowitz type of SA. In many practical problems, the gradient is not readily available. For example, in the traffic control application, one has no idea of the exact nature of the functional relationship of the loss (total delay time) to the parameters (signal light settings), much less the derivative of that function.

A standard approach to approximating the gradient is the "finite difference" method of elementary calculus. In this method, the *i*th component of the gradient approximation is computed as

$$\frac{y(\theta + ce_i) - y(\theta - ce_i)}{2c}, \tag{2}$$

where (as mentioned above) $y(\bullet)$ is the actual, possibly noisy, measurement of $L(\bullet)$; e_i is a vector with a one in the *i*th component and zero in all the other components; and c is a small positive number. Aficionados of numerical methods will recognize this as a "two-sided" gradient approximation. There is also a "one-sided" version which is sometimes used, but the distinction is not important for our purposes here. Using this gradient approximation in Eq. 1 results in an approximate-gradient SA in which the *i*th component of the gradient approximation is

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

where $\{c_k\}$ is usually chosen to be a decreasing sequence of small positive numbers. This form of SA, often called a finite difference SA (FDSA), is the classical form of the gradient-free SA for stochastic optimization. The FDSA has been extensively studied (e.g., Kushner and Yin,[3] chapters 1, 5, 6, and others; Spall,[4] chapters 6 and 7 and references therein) and is known to converge to a local minimum (see the discussion below) of the loss function under various conditions.

Simultaneous Perturbation Stochastic Approximation

The main focus of this article involves the SPSA algorithm, which was developed at APL by Jim Spall.[5] SPSA is an approximate-gradient SA that uses the standard algorithm form shown in Eq. 1 as well as a special form of gradient approximation called the "simultaneous perturbation" gradient approximation. In SPSA, the *i*th component of the gradient approximation is computed as

$$\frac{y(\theta + c\Delta) - y(\theta - c\Delta)}{2c\Delta_i}, \tag{4}$$

where Δ is a vector having the same size as θ and contains elements randomly generated by computer according to specifications,[5] and Δ_i is the *i*th component of Δ. The Δ_i are usually (but not necessarily) generated from the Bernoulli (± 1) distribution. Uniformly or normally distributed perturbations are *not* allowed by the conditions in Ref. 5. Substituting the gradient approximation shown in Eq. 4 into Eq. 1 results in the approximate-gradient SPSA in which the *i*th component of the gradient approximation is

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_{(k)}) - y(\hat{\theta}_k - c_k\Delta_{(k)})}{2c_k\Delta_{(k)i}}. \tag{5}$$

Here, $\Delta_{(k)}$ is the vector of random elements generated at iteration k, and $\Delta_{(k)i}$ is the *i*th component of $\Delta_{(k)}$. An excellent introductory discussion of SPSA is given in Ref. 6.

While the approximations in Eqs. 2 and 4 have a superficial similarity, they are in fact quite different computationally. Since this difference is important to our story, let us look more closely at the two approximations. The FDSA gradient approximation vector is

$$\begin{bmatrix} [y(\theta + ce_1) - y(\theta - ce_1)]/2c \\ [y(\theta + ce_2) - y(\theta - ce_2)]/2c \\ \vdots \\ [y(\theta + ce_p) - y(\theta - ce_p)]/2c \end{bmatrix}, \tag{6}$$

where the dimension of θ is p (i.e., θ is a p vector), while the SPSA gradient approximation is

$$\begin{bmatrix} [y(\theta + c\Delta) - y(\theta - c\Delta)]/2c\Delta_1 \\ [y(\theta + c\Delta) - y(\theta - c\Delta)]/2c\Delta_2 \\ \vdots \\ [y(\theta + c\Delta) - y(\theta - c\Delta)]/2c\Delta_p \end{bmatrix}. \quad (7)$$

Note that the FDSA gradient (Expression 6) needs to obtain the measurement $y(\bullet)$ of the loss function at $2p$ values of the argument $(\theta \pm ce_i)$, whereas the SPSA gradient (Expression 7) only needs *two* measurements of $y(\bullet)$, *regardless of the size (p) of* θ, since the components of the approximation in Expression 7 differ only by the Δ_i in the denominator. Since the Δ_i are computer-generated random numbers, the cost of obtaining the Δ_i is essentially zero. In contrast, the cost of obtaining the loss function measurements can be huge. For example, in the traffic control application, obtaining this measurement might require having a large group of people on the streets with stopwatches. So in terms of (possibly expensive) loss function measurements, the FDSA gradient approximation costs p times as much to compute as that of SPSA. This difference means that SPSA has the *potential* for requiring far fewer measurements than FDSA in finding θ^*. On the other hand, the FDSA computation tends (at least when the loss function is not corrupted by noise) to produce a very good gradient approximation since it follows the standard definition from calculus. In contrast, the SPSA gradient is not a very good approximation to the true gradient. This direct connection to the true gradient would seem to give FDSA a potential edge in finding θ^*. However, Spall[5] has shown that the SPSA potential advantage is the dominant one *for optimization*. In fact, under reasonably general conditions, SPSA can do about as well as FDSA through the use of only $1/p$ times the number of loss function measurements that FDSA would require.

An intuitive explanation of this remarkable performance advantage for SPSA is that the algorithm (shown in Eq. 1) naturally results in a kind of averaging (controlled by the a_k terms) of the gradient approximations *across iterations*. Since the SPSA gradient approximation is an "almost unbiased estimator" of the gradient,[6] the errors in the approximation tend to average out over the long run of iterations. This $1/p$ advantage for SPSA can result in immense savings for a complex (large p) application when the loss function measurements are costly to obtain.

This relative efficiency of SPSA has important implications for APL's work for DoD as well as civilian applications. For one thing, we have a powerful tool for projects that require optimization in a complex setting. In addition, problems that could not previously be attempted with conventional methods (because of the infeasibility of collecting the loss function measurements or impractical computation times) may now be solvable. Examples are the traffic control, optimal targeting, and munitions-locating projects mentioned above, which are discussed in more detail in Spall.[6]

SPSA has proven to be an efficient and effective optimization tool in many complex applications. In addition, a large body of literature on the theory and practice of SPSA has accumulated, generalizing the convergence properties of the basic form, introducing modified forms of the algorithm and proving their properties, giving instruction on how to set up and run the algorithm, and describing numerous successful applications. A good source of this information is the APL Web site http://www.jhuapl.edu/spsa.

GLOBAL OPTIMIZATION

Introduction

A common desire in many applications is that the algorithm will reach the best possible minimum rather than be left at a "local minimum" value of the parameter. A glance at Fig. 3 provides an intuitive idea of what we mean by this. The "minimum" of $L(\theta)$ at θ_a^* is called a local minimum, since all of the θ values in some small neighborhood of θ_a^* give a larger value of $L(\theta)$. However, θ_a^* does not give the best possible minimum of $L(\theta)$ available in the "allowable θ domain." The smallest value of $L(\theta)$ in this example is $L(\theta_b^*)$, and so θ_b^* is called the "global" minimum (θ_b^* is sometimes *also* referred to as a local minimum since it satisfies the definition, but we will avoid that usage). Figure 4 gives a slightly more sophisticated (since θ here is a two-vector) picture of a function $L(\theta)$ having both local and global minima. In "real life," it is not always obvious whether the situation exhibits both local and global minima, nor what the impact would be of using only a locally best solution. For example, in the traffic control scenario, we can speculate that the standard fixed traffic signal timing strategy might produce a local minimum of the average delay time, while police officers directing traffic at key intersections might produce the truly best result since they can more readily react to the traffic flow conditions at the moment. Of course, in an application where the loss function is directly related to resources or finances (e.g., timing trades in the stock market), the value of finding the global optimum solution may be more obvious.

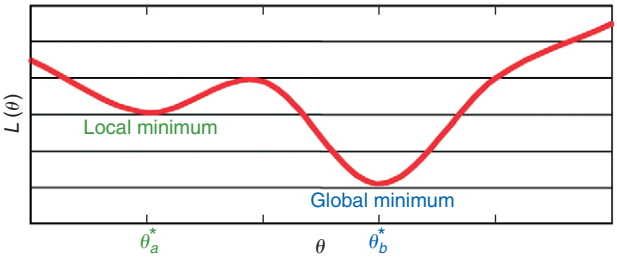

Figure 3. Local and global minima.

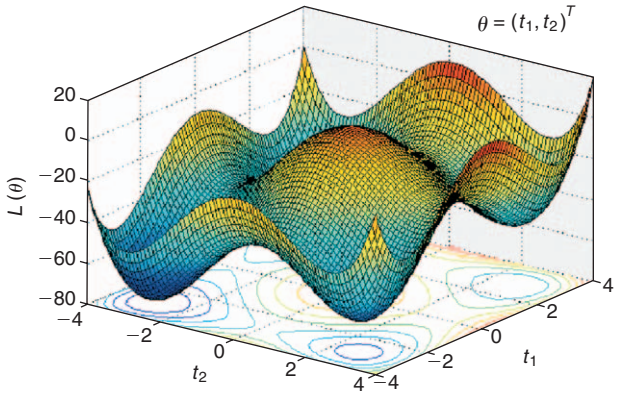

Figure 4. Local and global minima for a two-dimensional loss function. Unique global minimum is near $[-2.9, -2.9]^T$. (Reprinted from Ref. 4 with permission of John Wiley & Sons, Inc., © 2003.)

Incidentally, the idea of an allowable θ domain (mentioned above) is an important, and sometimes complex, issue in most applications. Usually there are practical limits on the θ values that can be considered. Indeed, sometimes θ cannot take on a continuum of values (e.g., one component of θ might be an integer enumerating the amount of some resource); in that case, gradient-based methods (approximate or not) may not be applicable. One result of this is that the algorithm may need to be specially designed to handle these constraints (see, e.g., Wang and Spall[7]). For simplicity in this article, we will ignore these complications and assume that the components of θ are all defined on a limitless continuum.

Also for simplicity, let's continue to assume just one global minimum of $L(\theta)$. Slightly more general results may include the assumption that there are a finite or infinite number of global minimum points θ_i^* such that $L(\theta)$ has the same value at every θ_i^* and for all other θ, $L(\theta) > L(\theta_i^*)$, i.e., strictly larger than at the global minima. This distinction is not important for our message here.

Using Injected Noise to Promote Global Optimization

Several authors[8–11] have examined the problem of global optimization using various forms of gradient-free SA. The usual approach involves using the computer to add random noise to an algorithm like Eq. 1, where $\hat{g}_k(\hat{\theta}_k)$ may be the FDSA gradient approximation or the actual gradient $g(\hat{\theta}_k)$. In the latter case, which may be referred to as "steepest descent with injected noise," the algorithm is

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k g(\hat{\theta}_k) + q_k \omega_k \,, \qquad (8)$$

where q_k are appropriately selected scalars and ω_k are random (usually standard Gaussian) p-dimensional vectors satisfying certain conditions. It is known that carefully injecting noise in this way can result in an algorithm that converges (in some sense) to the global minimum. For a discussion of the conditions, results, and proofs, see, e.g., Gelfand and Mitter,[9] Kushner,[10] and Fang et al.[12] These results are based on the intuitive idea that promoting global convergence by the injection of extra noise terms into the recursion may allow the algorithm to escape θ neighborhoods that produce local minimum points of $L(\theta)$, especially in the early iterations of the algorithm. This idea is illustrated in Fig. 5. The amplitude of the injected noise is decreased over time (a process called "annealing") so that the algorithm can finally converge when it reaches the neighborhood of the global minimum point.

GLOBAL OPTIMIZATION USING SPSA

Overview

The main goal of this article is to discuss some global convergence properties of SPSA. Perhaps this is a good point to mention the value of *theoretical* results in an applied science like optimization. An analyst with an optimization problem is, of course, interested in using an algorithm that will yield a solution that improves the performance of the actual (physical) system. Since there are many algorithms available, the analyst may look at various factors to help in the selection of an algorithm such as:

- Numerical studies or recommendations of other practitioners. These may be useful if the applications are similar to the scenario of interest to the analyst. However, the results of an algorithm or study in an unrelated application are usually not meaningful in themselves. In fact, optimization theory provides confirmation of this common-sense idea in the form of "no-free-lunch theorems," which prove essentially that no algorithm will work well on all problems.

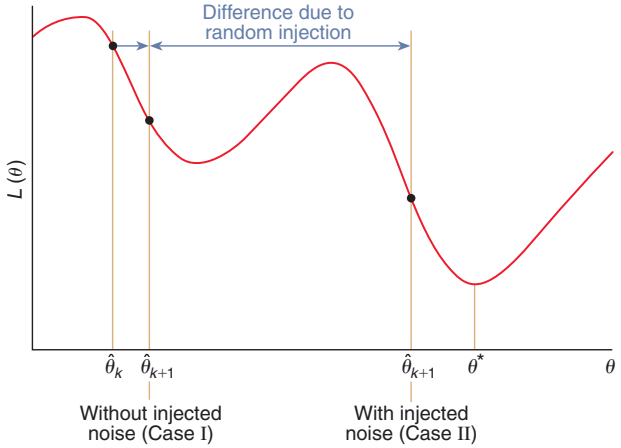

Figure 5. Injection of random noise can promote convergence to the global minimum of $L(\theta)$. Case I: $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k)$. Case II: $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k)$ + injected noise.

- Intuitive notions that an algorithm might work well. These approaches, sometimes based on an analogy with a successful physical or biological system, can seem reasonable, but it may be difficult to establish that they are useful in a specific application.
- Mathematical results that describe the algorithm's performance under various hypotheses. A difficulty with such theoretical results is that the hypotheses may not be fully satisfied (or may be hard to verify) in a real-world application. Furthermore, such results tend to refer to the asymptotic (large k) performance of the algorithm. But analysts often prefer an algorithm that is proven to perform well (e.g., converges to the correct optimum value and exhibits a certain rate of convergence) under some conditions, compared to an algorithm that has little or no theoretical support. In addition, asymptotic effects often are seen after a reasonable number of iterations, and theory often can serve as an indication of the *type of problem* in which an algorithm will perform well.

A considerable body of theory has been developed for SPSA, for example, Spall (chapter 7),[4] Spall,[5,13] Chin,[14] Dippon and Renz,[15] and the SPSA Web site given previously. Prior to the work that we report here, this theory did not include global convergence results. Recall that global convergence theory does exist for standard implementations of SA. However, because of the particular form of SPSA's gradient approximation, existing theory on the global convergence of standard SA algorithms is not directly applicable to SPSA. In the next section we discuss a theorem showing that SPSA can achieve global convergence ("in probability"; see the definition in Comment (a) after Theorem 1 below) by the technique of injecting noise. The convergence-in-probability results of our Theorem 1 and Theorem 2 are standard types of global convergence results. Several authors have shown or discussed global convergence in probability or in distribution.[9,12,16–21] Stronger "almost sure" global convergence results seem only to be available by using a generally infeasible exhaustive search[22] or random search methods,[23] or for cases of optimization, in a discrete θ space.[24]

In the subsequent section, we discuss a theorem showing that SPSA can, under different conditions, achieve global convergence without the injection of extra noise. As will be seen, this can have advantages in setting up the algorithm and can provide a dramatic increase in the speed of convergence relative to classical approximate-gradient SA algorithms that need injected noise.

SPSA with Injected Noise

Our first theorem applies to the following algorithm, which is the basic SPSA recursion indicated in Eq. 1, modified by the addition of extra noise terms:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k) + q_k \omega_k \,, \tag{9}$$

where

ω_k = (usually) a p-dimensional vector of independent, identically distributed (i.i.d.) standard Gaussian injected noise,

$a_k = a/k$,

$q_k^2 = q / k \log\log(k)$,

$a > 0$,

$q > 0$, and

$\hat{g}_k(\bullet)$ = the simultaneous perturbation gradient defined in Eqs. 4 and 5.

This is similar to Eq. 8, but here the SPSA gradient approximation replaces the true gradient.

Our Theorem 1 (below), on the global convergence of SPSA using injected noise, is based on a result in Gelfand and Mitter.[9] The theorem requires eight technical hypotheses, which we will not list here. They can be found in Maryak and Chin.[25] The hypotheses include

- Descriptions of and restrictions on the SPSA setup (coefficients, choice of Δ_k, etc.)
- Conditions on the measurement noise attached to $L(\theta)$
- Conditions on the injected noise terms
- Conditions on the loss function, mainly differentiability and boundedness conditions
- A boundedness condition on the iterates $\hat{\theta}_k$

We can now state the first important result of this article as follows:

Theorem 1: Under the hypotheses discussed above, $\hat{\theta}_k$ (in Eq. 9) converges in probability to the global minimum θ^*.

Comments

(a) Convergence "in probability" is a standard type of convergence involving sequences of random variables. It means that, for any $\rho > 0$, a certain probability converges to zero as k approaches infinity, i.e., $\Pr\{|\hat{\theta}_k - \theta^*| > \rho\} \to 0$ as $k \to \infty$.

(b) The proof of this theorem actually illustrates a little more than stated here, since it shows convergence in probability to a possible *set* of global minima. That is, it allows for a finite set of values θ_i^*, each giving the same value of $L(\theta_i^*)$, which is smaller than $L(\theta)$ at all other values of θ.

(c) The proof is an application of Theorem 2 in Gelfand and Mitter[9] and can be seen in Maryak and Chin.[26]

SPSA Without Injected Noise

As indicated previously in the Overview, the injection of noise into an algorithm, while providing for

global optimization, introduces some difficulties such as the need for more "tuning" (e.g., selecting the coefficients) of the extra terms q_k and ω_k, and retarded convergence in the vicinity of the solution, which is due to the continued addition of noise. Also, the definition of the SPSA approximate gradient (discussed above) offers an intuitive reason to suspect that SPSA without the addition of extra noise may act somewhat like a standard SA algorithm having injected noise. We will expand on these ideas in the subsequent discussion.

First, we present Theorem 2, which states the main result of this article: that *basic* SPSA (i.e., *without* injected noise) does indeed achieve the same type of global convergence as in Theorem 1, but under a different set of conditions. Theorem 2 is based on 12 hypotheses, many of which are similar to those of Theorem 1. The major differences in conditions involve some further conditions on the iterates $\hat{\theta}_k$, and on an ordinary differential equation $\partial\theta/\partial t = g(\theta(t))$ based on the gradient of the loss function, where $\theta(t)$ is an extension of the (discrete-time) iterates to a continuous-time function. These hypotheses can be seen in Maryak and Chin.[25] Although the conditions are quite technical and difficult to check in practice, they are standard forms that are familiar to specialists.

Let us emphasize here that we are working with the basic SPSA algorithm having the same form as Eq. 1:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k), \tag{10}$$

where $\hat{g}_k(\bullet)$ is the simultaneous perturbation approximate gradient defined in Eqs. 4 and 5, and now (obviously) no extra noise is injected into the algorithm. Now we can state our main theorem:

Theorem 2: Under the 12 hypotheses discussed above, $\hat{\theta}_k$ in Eq. 10 converges in probability to the global minimum θ^*.

Comments

(a) As in Theorem 1, the proof of Theorem 2 actually shows convergence in probability to the *set* of global minima of $L(\theta)$.

(b) The details of the proof of our Theorem 2 are available from the authors. The idea of the proof is as follows. This theorem is based on a result in Kushner[10] in which he discusses an algorithm $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k[g(\hat{\theta}_k) + \zeta_k]$, where ζ_k is i.i.d. Gaussian (injected) noise. To prove our Theorem 2, we start by writing the SPSA recursion as $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k[g(\hat{\theta}_k) + \zeta_k^*]$, where $\zeta_k^* \equiv \hat{g}_k(\hat{\theta}_k) - g(\hat{\theta}_k)$ is the "effective noise" (relative to the true gradient, as discussed in Comment (c) below) introduced by the inaccuracy of the SPSA gradient approximation. So, our algorithm has the same form as that in Kushner.[10] However, since ζ_k^* is not i.i.d. Gaussian, we cannot use Kushner's result directly. Instead, we use material in Kushner and Yin[3] to establish a key "large deviation" result related to our algorithm in Eq. 10, which allows the global convergence conclusion in Kushner[10] to be used with ζ_k^* replacing the ζ_k in his algorithm.

(c) The definition of the SPSA gradient approximation provides some intuition on why Theorem 2 is possible, i.e., on why SPSA might not need to use injected noise for global convergence. As discussed above, although the SPSA gradient approximation tends to work very well in an SA recursion, the simultaneous perturbation gradient, evaluated at any single point in θ space, tends to be a poor estimate of the true gradient evaluated at θ. One is therefore led to consider whether the *effective* noise introduced (automatically) into the recursion by this inaccuracy is sufficient to provide for global convergence *without* a further injection of additive noise. This idea can be expressed as follows:

- The SPSA gradient approximation = the true gradient + SPSA-induced error.
- Therefore, the SPSA algorithm = the steepest descent algorithm + "SPSA-induced noise."
- We know that the steepest descent algorithm + ("carefully selected noise") will, under the proper conditions, converge to the global minimum of the loss function.
- Now, if the SPSA-induced noise acts enough like the carefully selected noise, then the SPSA algorithm (with no extra noise) might also converge to the global minimum.

It turns out that the SPSA-induced noise is, in general, not the same as the carefully selected noise (for example, the i.i.d. Gaussian requirement is typically not met), and a formal proof of Theorem 2 was needed.

Importance of Theorem 2: Rate of Convergence

Theorem 2 describes the global convergence of SPSA without the addition of injected noise. This is important because the use of injected noise in such an algorithm can have a significant negative effect on the algorithm's rate of convergence performance. Let us examine this phenomenon in a little more detail. A useful way to study the rate of convergence is via asymptotic (i.e., large k) convergence results (similar to the "law of large numbers"), by which it can be seen[25] that, for large k, the ratio of the error, $(\hat{\theta}_k - \theta^*)$, for the algorithm with injected noise to that of the algorithm without injected noise is proportional to $k^{1/3}/\sqrt{\log\log(k)}$. Figure 6 shows a plot of this ratio versus the iteration number

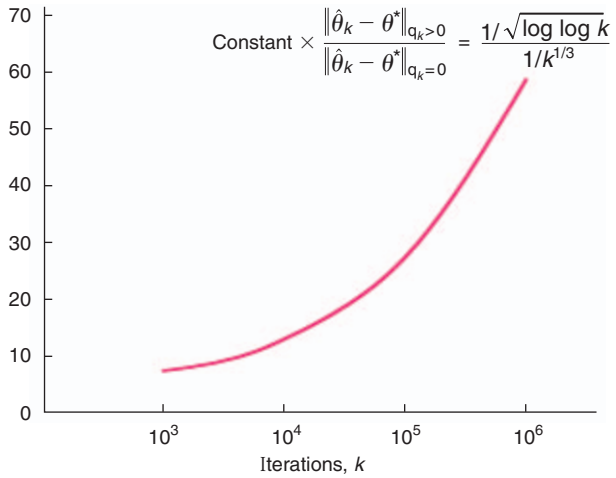

Figure 6. Ratio of large-sample estimate errors with (q_k>0) and without (q_k=0) injected randomness (modified from http://www.jhuapl.edu/ISSO/ with permission of the author, Jim Spall).

k, giving an idea of the difference in convergence rates possible in these two versions of the SPSA algorithm. The figure indicates (for example) that, at 100,000 iterations, the algorithm without injected noise is converging about 30 times faster than the algorithm with injected noise. This advantage for the non–injected-noise algorithm increases as the number of iterations increases, at a rate roughly proportional to $k^{1/3}$ (since the "root log log" term changes very slowly). This dramatic difference in rate of convergence provided strong motivation to establish a global convergence result for SPSA without injected noise.

NUMERICAL STUDY: SPSA WITHOUT INJECTED NOISE VERSUS A GENETIC ALGORITHM

To test the global performance of SPSA, we applied SPSA to a loss function given in Example 6 of Styblinski and Tang[11]:

$$L(\theta) = (2p)^{-1}\sum_{i=1}^{p} t_i^2 - 4p\prod_{i=1}^{p}\cos(t_i)\,, \qquad (11)$$

where $p = 5$ and $t_1, \ldots, t_5$ are the components of θ. This function has the global minimum value of −20 at the origin and a large number of local minima. Our goal is to compare the performance of SPSA without injected noise to a genetic algorithm (GA). GAs are intuitively based on biological evolutionary processes and seek to emulate the optimization effects found in nature. These algorithms have been popular for many years and have often been applied with success in optimization applications where global convergence is a concern; see Spall[4] (chapters 9 and 10) and Mitchell.[27]

We implemented a GA using the popular features of elitism (elite members of the old population pass unchanged into the new population), tournament selection (tournament size = 2), and real-number encoding (see Mitchell,[27] pp. 168, 170, and 157, respectively). We used the following settings for the GA algorithm. The population size was 80, the number of elite members (those carried forward unchanged) in each generation was 10, the crossover rate was 0.8, and mutation was accomplished by adding a Gaussian random variable with mean zero and standard deviation 0.01 to each component of the offspring. All runs of the GA algorithm reported here used 1000 evaluations of the loss function.

We experimented with these settings to try to enhance the performance of the GA algorithm. In particular, the performance of this GA did not change much in runs that used up to 5000 evaluations of the loss function. The original population of 80 (five-dimensional) θ vectors was created by uniformly randomly generating points in the five-dimensional hypercube centered at the origin, with edges of length 6 (so that all components had absolute value less than or equal to 3 rad). Of course, the GA actually computes with what is often called a "fitness" function (which it tries to maximize), which in this example is $-L(\theta)$. The best loss function value found by the algorithm in each of the 10 independent runs of GA is shown in Table 1. Although the algorithm did reasonably well in getting close to the minimum loss value of −20, it did not find the global minimum in any of the 10 runs.

We examined the performance of basic SPSA (without adding injected noise) using the algorithm parameters $a_k = a/(k + A)^{\alpha}$ and $c_k = c/k^{\gamma}$, with $A = 20$, $a = 0.5$, $\alpha = 0.602$, $c = 0.5$, and $\gamma = 0.101$. For each run of SPSA, we started θ at a point randomly chosen in the same hypercube mentioned above, and we did not constrain the search space for SPSA or for GA. We ran 10 Monte Carlo trials (randomly varying the starting point and the choices of Δ_k). The SPSA algorithm converged within 1000 evaluations of the loss function in 5 out of the 10 cases. In most of the other cases, the algorithm appeared to get stuck at a local minimum of −18.05 (Table 1). The difference between the two average values shown in Table 1 is statistically significant (T-test @ 5%). The loss function used in this study was chosen for two reasons: (1) the correct answer is known (which is often not the case in "real-life" problems), and (2) it offers a demanding test of the algorithm, as evidenced by the fact that SPSA converged to one of the many *local* minima in some of its trials and the GA did so in all of its trials.

The results of this numerical study show a reasonably good performance of the basic SPSA algorithm in this difficult global optimization problem. Another numerical study demonstrating good global convergence of basic SPSA compared to a GA and to another popular algorithm (simulated annealing) often used to promote

Table 1. Final loss function value in each of 10 independent runs of two algorithms.

Run	SPSA	GA
1	−18.05	−11.52
2	−20.00	−15.64
3	−16.09	−13.61
4	−18.05	−13.84
5	−18.05	−17.10
6	−20.00	−15.55
7	−20.00	−17.85
8	−20.00	−17.21
9	−18.05	−17.21
10	−20.00	−16.02
Average value	−18.83	−15.56
Number of function evaluations	1000	1000

global convergence is described in Maryak and Chin.[25] Further details on GAs and a study in which a GA out-performed SPSA (with injected noise) are given in Spall[4] (chapter 9).

SUMMARY

SPSA is an efficient gradient-free SA algorithm that has performed well on a variety of complex optimization problems. The work reported here has established that, as with some standard SA algorithms, adding injected noise to the basic SPSA algorithm can result in a global optimizer. More significantly, we showed that, under certain conditions, the basic SPSA recursion can achieve global convergence *without the need for injected noise*. The use of basic SPSA as a global optimizer can ease the implementation of the global optimizer (no need to tune the injected noise) and result in a significantly faster rate of convergence (no extra noise corrupting the algorithm in the vicinity of the solution). In the numerical studies, basic SPSA demonstrated good performance as a global optimizer, often finding the global minimum of a very tricky five-dimensional loss function having many local minima.

REFERENCES

[1]Reardon, B. E., *Description of Stochastic Optimization Toolbox, v1.0*, A1E(04)U-4-005, JHU/APL, Laurel, MD (20 Feb 2004).

[2]Reardon, B. E., Palumbo, N. F., and Casper, S. G., "Simulation-Based Performance Optimization of Missile Guidance and Control Algorithms," in *Proc. 11th Ann. AIAA/MDA Technology Conf. and Exhibit*, Williamsburg, VA (29 Jul–2 Aug 2002).

[3]Kushner, H. J., and Yin, G. G., *Stochastic Approximation and Recursive Algorithms and Applications*, Springer, New York (2003).

[4]Spall, J. C., *Introduction to Stochastic Search and Optimization*, John Wiley & Sons, Hoboken, NJ (2003).

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

[6]Spall, J. C., "An Overview of the Simultaneous Perturbation Method for Efficient Optimization," *Johns Hopkins APL Tech. Dig.* **19**(4), 482–492 (1998).

[7]Wang, I.-J., and Spall, J. C., "Stochastic Optimization with Inequality Constraints Using Simultaneous Perturbations and Penalty Functions," in *Proc. IEEE Conf. on Decision and Control*, Maui, HI, pp. 3808–3813 (9–12 Dec 2003).

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

[9]Gelfand, S. B., and Mitter, S. K., "Recursive Stochastic Algorithms for Global Optimization in $\mathbf{R}^d$," *SIAM J. Control Optim.* **29**, 999–1018 (1991).

[10]Kushner, H. J., "Asymptotic Global Behavior for Stochastic Approximation and Diffusions with Slowly Decreasing Noise Effects: Global Minimization via Monte Carlo," *SIAM J. Appl. Math.* **47**, 169–185 (1987).

[11]Styblinski, M. A., and Tang, T.-S., "Experiments in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing," *Neural Net.* **3**, 467–483 (1990).

[12]Fang, H., Gong, G., and Qian, M., "Annealing of Iterative Stochastic Schemes," *SIAM J. Control Optim.* **35**, 1886–1907 (1997).

[13]Spall, J. C., "Adaptive Stochastic Approximation by the Simultaneous Perturbation Method," *IEEE Trans. Automat. Control* **45**, 1839–1853 (2000).

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

[15]Dippon, J., and Renz, J., "Weighted Means in the Stochastic Approximation of Minima," *SIAM J. Control Optim.* **35**, 1811–1827 (1997).

[16]Chaing T-S., Hwang, C-R., and Sheu, S-J., "Diffusion for Global Optimization in $\mathbf{R}^n$," *SIAM J. Control Optim.* **25**, 737–753 (1987).

[17]Gelfand, S. B., and Mitter, S. K., "Metropolis-Type Annealing Algorithms for Global Optimization in $\mathbf{R}^d$," *SIAM J. Control Optim.* **31**, 110–131 (1993).

[18]Geman, S., and Geman, D., "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Trans. Pattern Anal. Machine Intel.* **PAMI-6**, 721–741 (1984).

[19]Hajek, B., "Cooling Schedules for Optimal Annealing," *Math. Oper. Res.* **13**, 311–329 (1988).

[20]Yakowitz, S., L'Ecuyer, P., and Vazquez-Abad, F., "Global Stochastic Optimization with Low-Dispersion Point Sets," *Operations Res.* **48**, 939–950 (2000).

[21]Yin, G., "Rates of Convergence for a Class of Global Stochastic Optimization Algorithms," *SIAM J. Optim.* **10**, 99–120 (1999).

[22]Dippon, J., and Fabian, V., "Stochastic Approximation of Global Minimum Points," *J. Statist. Plan. Inference* **41**, 327–347 (1994).

[23]Yakowitz, S., "A Globally Convergent Stochastic Approximation," *SIAM J. Control Optim.* **31,** 30–40 (1993).

[24]Alrefaei, M. H., and Andradottir, S., "A Simulated Annealing Algorithm with Constant Temperature for Discrete Stochastic Optimization," *Management Sci.* **45**, 748–764 (1999).

[25]Maryak, J. L., and Chin, D. C., "Global Random Optimization by Simultaneous Perturbation Stochastic Approximation," in *Proc. Am. Control Conf.*, Arlington, VA, pp. 756–762 (Jun 2001).

[26]Maryak, J. L., and Chin, D. C., "Efficient Global Optimization Using SPSA," in *Proc. Am. Control Conf.*, San Diego, CA, pp. 890–894 (1999).

[27]Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA (1996).

ACKNOWLEDGMENT: This work was supported by the APL Independent Research and Development Program.

THE AUTHORS

JOHN L. MARYAK received a Ph.D. in mathematics from the University of Maryland in 1972. He has worked at APL since 1977 on diverse tasks involving the analysis and performance assessment of complex military systems, mainly the Navy's Trident submarine systems. This work has involved methodology development, software development, planning and coordination of studies and analyses, and leading projects for the statistical analysis and reporting of the performance of these systems. In addition, Dr. Maryak has participated in several IR&D projects on statistical theory and methodology and has published a number of papers on statistical methods and mathematical modeling. His e-mail address is john.maryak@jhuapl.edu.

DANIEL C. CHIN is a Senior Professional Staff mathematician who has worked at APL in the Strategic Systems Department for more than 20 years. He received a B.S. in mathematics from Chung Yuan University, Taiwan, in 1966 and an M.S. in mathematics from Northern Illinois University in 1970. Mr. Chin has primarily worked as a weapon system accuracy evaluator for the Navy's Trident system. He is experienced in stochastic approximation, data fusion, Bayesian analysis, statistical estimation and simulation, and image data processing. He has also participated in IR&D projects in statistical theory and methods, adaptive traffic control, and discrimination analysis for buried objects. Mr. Chin has published a number of papers on statistical estimation methodology, mathematical modeling, system-wide traffic control, and discrimination analysis. He is a member of the American Statistical Association (ASA), the Institute of Electrical and Electronics Engineers (IEEE), and Sigma Xi. He was on the team that won the Hart Prize for the most outstanding IR&D project at APL in 1990. His e-mail address is daniel.chin@jhuapl.edu.