Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization

JAMES C. SPALL, Senior Member, IEEE
The Johns Hopkins University

The need for solving multivariate optimization problems is pervasive in engineering and the physical and social sciences. The simultaneous perturbation stochastic approximation (SPSA) algorithm has recently attracted considerable attention for challenging optimization problems where it is difficult or impossible to directly obtain a gradient of the objective function with respect to the parameters being optimized. SPSA is based on an easily implemented and highly efficient gradient approximation that relies on measurements of the objective function, not on measurements of the gradient of the objective function. The gradient approximation is based on only two function measurements (regardless of the dimension of the gradient vector). This contrasts with standard finite-difference approaches, which require a number of function measurements proportional to the dimension of the gradient vector. This paper presents a simple step-by-step guide to implementation of SPSA in generic optimization problems and offers some practical suggestions for choosing certain algorithm coefficients.

Manuscript received August 14, 1996; revised September 24 and October 16, 1997.

IEEE Log No. T-AES/34/3/06015.

This work was partially supported by U.S. Navy Contract N00024-97-C-8119 and the JHU/APL Independent Research and Development Program.

Author's address: The Johns Hopkins University, Applied Physics Laboratory, Johns Hopkins Rd., Laurel, MD 20723-6099; E-mail: (james.spall@jhuapl.edu).

I. BACKGROUND

Stochastic optimization algorithms are used in virtually all areas of engineering and the physical and social sciences. Such techniques apply in the usual case where a closed-form solution to the optimization problem of interest is not available and where the input information into the optimization method (e.g., loss function evaluations) may be contaminated with noise. Typical applications include model fitting and statistical parameter estimation, experimental design, adaptive control, pattern classification, simulation-based optimization, and performance evaluation from test data. Frequently, the solution to the optimization problem corresponds to a vector of parameters at which the gradient of the objective (say, loss) function with respect to the parameters being optimized is zero. In many practical settings, however, the gradient of the loss function for use in the optimization process is not available or is difficult to compute (knowledge of the gradient usually requires complete knowledge of the relationship between the parameters being optimized and the loss function). So, there is considerable interest in techniques for optimization that rely on measurements of the loss function only, not on measurements (or direct calculations) of the gradient (or higher order derivatives) of the loss function.

One of the techniques using only loss function measurements that has attracted considerable recent attention for difficult multivariate problems is the simultaneous perturbation stochastic approximation (SPSA) method introduced in Spall [26] and more fully analyzed in Spall [27]. Recent applications are described, e.g., in Hill and Fu [12, 13] (queueing systems), Rezayat [21] (industrial quality improvement), Maeda, et al. [16] (pattern recognition), Cauwenberghs [4] (neural network training), Spall and Cristion [31, 32] (adaptive control of dynamic systems), Alessandri and Parasini [1] (statistical model parameter estimation/fault detection), Sadegh and Spall [24] (sensor placement and configuration), Nechyba and Xu [18] (human-machine interface control), and Chin and Smith [7] (traffic management). SPSA is based on a highly efficient and easily implemented "simultaneous perturbation" approximation to the gradient: this gradient approximation uses only *two* loss function measurements independent of the number of parameters (say, p) being optimized. This contrasts, for example, with the standard (two-sided) finite-difference approximation (e.g., Dennis and Schnabel [8]), which uses $2p$ function measurements to approximate the gradient. The well-known Kiefer–Wolfowitz [14] SA algorithm (Ruppert [22]) is based on the finite-difference gradient approximation. The fundamental (and perhaps surprising) theoretical result in Spall [27, sect. 4] is:

Under reasonably general conditions, SPSA and Kiefer–Wolfowitz finite-difference-based SA (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 (since each gradient approximation uses only $1/p$ the number of function evaluations).

This theoretical result has been confirmed in many numerical studies by the author and others (see references), even in cases where p is on the order of several hundred or thousand.

The genesis for this paper is the many requests the author has received for a "cookbook" type guide to implementation of SPSA. Existing documentation (such as cited above) has tended to focus on specific applications or on the methodological, theoretical, and numerical properties of the algorithm without dwelling on the basics of practical implementation for general problems. Since the focus here is on implementation (and since much other documentation is readily available covering other aspects), this paper is not intended as a stand-alone document for motivating and demonstrating SPSA. In addition to the references cited above, the reader may wish to consult Chin [6], Wang and Chong [34], and Spall [28] for such information. It is hoped that this paper will help the potential user in developing software and choosing the kind of application-specific algorithm parameters (e.g., algorithm gain sequences) that *all* stochastic optimization algorithms (including techniques such as simulated annealing, genetic algorithms, evolutionary search, adaptive random search, etc.) need.

Since we are considering the *stochastic* optimization context (allowing for noisy measurements of the objective function), it is difficult to make general statements about the relative efficiency of SPSA with well-known *deterministic* algorithms that are oriented to the case of perfect measurements of the objective function (such as the many variations of the Nelder–Mead [19] algorithm). It is even more difficult to make meaningful comparisons with algorithms such as conjugate gradient and Newton–Raphson since those deterministic algorithms require direct gradient information. Such comparisons must be done on a case-by-case basis, reflecting the relative cost of obtaining the information required by the competing algorithms with the cost of obtaining a (possibly noisy) measurement of just the objective function. The above-mentioned comparison with finite-difference methods (FDSA) was performed on an equal basis of using the same type of information for both algorithms.

Section II summarizes the problem setting and introduces some notation. Section III presents a step-by-step guide to implementation that is aimed at helping the reader code the algorithm for his or her specific application. This section also offers suggestions regarding the choice of algorithm gain sequences. These suggestions were developed based on many test cases conducted by the author and others and form a reasonable basis if one has no specific reason to follow other guidelines. Since all stochastic optimization methods require the choice of such problem-specific algorithm coefficients, the gain selection issue for SPSA is not unique. Section IV summarizes some additional implementation aspects and related extensions of the basic SPSA algorithm.

II. BASIC ASSUMPTIONS AND FORMULATION

The goal is to minimize a loss function $L(\theta)$, where the loss function is a scalar-valued "performance measure" and θ is a continuous-valued p-dimensional vector of parameters to be adjusted. The SPSA algorithm works by iterating from an initial guess of the optimal θ, where the iteration process depends on the above-mentioned highly efficient "simultaneous perturbation" approximation to the gradient $g(\theta) \equiv \partial L(\theta)/\partial\theta$.

Assume that measurements $y(\theta)$ of the loss function are available at any value of θ:

$$y(\theta) = L(\theta) + noise.$$

For example, in a Monte Carlo simulation-based optimization context, $L(\theta)$ may represent the mean response with input parameters θ, and $y(\theta)$ may represent the outcome of one simulation experiment at θ. In some problems, exact loss function measurements will be available. This corresponds to the *noise* = 0 setting (and in the simulation example, would correspond to a deterministic—non-Monte Carlo—simulation). Note that no direct measurements (with or without noise) of the gradient are assumed available. This measurement formulation is identical to that of the FDSA algorithm discussed above and most implementations of genetic optimization algorithms and simulated annealing. It differs from the Robbins–Monro SA algorithm (which includes algorithms such as backpropagation for neural networks [White [35]] and infinitesimal perturbation analysis for discrete-event systems [Fu [11]] as special cases), Newton–Raphson search, and maximum likelihood scoring, all of which require direct measurement or calculation of $g(\theta)$.

It is assumed that $L(\theta)$ is a differentiable function of θ and that the minimum point θ^* corresponds to a zero point of the gradient, i.e.,

$$g(\theta^*) = \left.\frac{\partial L(\theta)}{\partial\theta}\right|_{\theta=\theta^*} = 0. \tag{1}$$

In cases where more than one point satisfies (1), then the algorithm may only converge to a local minimum (as a consequence of the basic recursive form of the algorithm there is generally not a risk of converging to a maximum or saddlepoint of

$L(\theta)$, i.e., to nonminimum points where $g(\theta)$ may equal zero). See Section IV below for a mention of modifications to the basic SPSA algorithm to allow it to search for the global solution among multiple local solutions. Note also that (1) is generally associated with unconstrained optimization; however, through the application of penalty functions and/or projection methods, it is possible to use (1) in a constrained problem (i.e., one where the θ values are not allowed to obtain certain values, usually as specified through equality and inequality constraints on the values of θ or $L(\theta)$). Section IV briefly discusses this further.

III. IMPLEMENTATION OF SPSA

Subsection IIIA presents the step-by-step guide to implementation and Subsection IIIB offers some suggestions for picking the algorithm coefficients (the SA gain sequences) to achieve reasonable practical performance.

A. Step-by-Step Implementation

The step-by-step summary below shows how SPSA iteratively produces a sequence of estimates. The Appendix presents an implementation of the steps below in MATLAB® code.

Step 1 *Initialization and Coefficient Selection.* Set counter index $k = 0$. Pick initial guess $\hat{\theta}_0$ and nonnegative coefficients a, c, A, α, and γ in the SPSA gain sequences $a_k = a/(A + k + 1)^\alpha$ and $c_k = c/(k + 1)^\gamma$. Practically effective (and theoretically valid) values for α and γ are 0.602 and 0.101, respectively (the asymptotically optimal values of 1.0 and 1/6 may also be used); a, A, and c may be determined based on the practical guidelines given in Subsection IIIB.[1]

Step 2 *Generation of 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 conditions in Spall [27]. A simple (and theoretically valid) choice for each component of Δ_k is to use a Bernoulli ± 1 distribution with probability of $\frac{1}{2}$ 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 $g(\hat{\theta}_k)$:

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

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.[2]

Step 5 *Updating θ Estimate.* Use the standard SA form

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

to update $\hat{\theta}_k$ to a new value $\hat{\theta}_{k+1}$.

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 e.g., in Pflug [20, pp. 297–300]).

B. Choice of Gain Sequences a_k, c_k

This subsection summarizes some additional implementation aspects regarding the choice of algorithm gain sequences. The reader should be warned that the guidelines provided here are just that—guidelines—and may not be the best for every application. These guidelines were developed based on many test cases conducted by the author and others and form a reasonable basis for starting if one has no specific reason to follow other guidelines (theoretical guidelines, such as discussed in Fabian [10], Chin [6], and Dippon and Renz [9] are not generally useful in practical applications since they require the very

[1] 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. Section IV below discusses a second-order version of SPSA that automatically (asymptotically) scales for different magnitudes.

[2] Despite the expense of the additional function evaluations, it is sometimes useful to average several SP gradient approximations (each with an independent value of Δ_k) at a given $\hat{\theta}_k$. This is especially the case when the noise levels in the $L(\theta)$ evaluations are high. Such averaging can often mitigate the fact that SPSA may be more unstable in the early iterations than FDSA due to its potentially poorer quality gradient approximation. In fact, even a relatively low amount of averaging (say, two to four SP gradient estimates) can sometimes make SPSA *more* stable than FDSA due to the reduced effective noise contributions. In high-dimensional systems, this averaged SP gradient estimate will still take many fewer measurements than FDSA. Theoretical justification for net improvements to efficiency by such gradient averaging is given in Spall [27].

information on the loss function and its gradients that is assumed unavailable!). However, it is likely that in serious applications, most users will want to refine the gain selection from that recommended here.

The choice of the gain sequences (a_k and c_k) is critical to the performance of SPSA (as with *all* stochastic optimization algorithms and the choice of their respective algorithm coefficients). With α and γ as specified in Step 1, one typically finds that in a high-noise setting (i.e., poor quality measurements of $L(\theta)$) it is necessary to pick a smaller a and larger c than in a low-noise setting. Although the asymptotically optimal values of α and γ are 1.0 and 1/6, respectively (Fabian [10] and Chin [6]), it appears that choosing $\alpha < 1.0$ usually yields better finite-sample performance through maintaining a larger step size; hence the recommendation in Step 1 to use values (0.602 and 0.101) that are effectively the lowest allowable satisfying the theoretical conditions mentioned in Section III (from Spall [27, prop. 2]). In a setting where a large amount of data are likely to be available, it may be beneficial to convert to $\alpha = 1.0$ and $\gamma = 1/6$ at some point in the iteration process to take advantage of their asymptotic optimality.

As a rule-of-thumb (with the Bernoulli ± 1 distribution for the elements of Δ_k as suggested in Step 1), it is effective to set c at a level approximately equal to the standard deviation of the measurement noise in $y(\theta)$ in order to keep the p elements of $\hat{g}_k(\hat{\theta}_k)$ from getting excessively large in magnitude (the standard deviation can be estimated by collecting several $y(\theta)$ values at the initial guess $\hat{\theta}_0$; a precise estimate is not required in practice). In the case where one had perfect measurements of $L(\theta)$, then c should be chosen as some small positive number.

The values of a, A can be chosen together to ensure effective practical performance of the algorithm. (The constant A is not typically shown in the SA literature, but we have found that including this "stability constant" is effective in allowing for a more aggressive, i.e., larger, a in the numerator by avoiding instabilities in the early iterations. A larger a may enhance performance in the later iterations by producing a larger step size when the effect of A is small.) A guideline we have found useful is to choose A such that it is much less than the maximum number of iterations allowed or expected, e.g., we frequently take it to be 10% (or less) of the maximum number of expected/allowed iterations and choose a such that $a/(A+1)^{\alpha}$ times the magnitude of elements in $\hat{g}_0(\hat{\theta}_0)$ is approximately equal to the smallest of the desired change magnitudes among the elements of θ in the early iterations. To do this reliably may require several replications of $\hat{g}_0(\hat{\theta}_0)$. For example, if it were felt that elements of θ should typically move by a magnitude 0.1 in the early iterations, and the magnitude of the elements in $\hat{g}_0(\hat{\theta}_0)$ (after choosing c and γ as above) is approximately 10, then with $A = 100$ and $\alpha = 0.602$, we would choose $a = 0.16$. These guidelines for choosing a are similar to those mentioned in Brennan and Rogers [3, sect. 2]. If the elements of θ vary greatly in magnitude, then these guidelines can be modified in an obvious way to accommodate a scaling matrix applied to a_k (see footnote 1).

IV. FURTHER RESULTS AND EXTENSIONS TO BASIC SPSA ALGORITHM

Sadegh and Spall [25] consider the problem of choosing the best distribution for the Δ_k vector. Based on asymptotic distribution results, it is shown that the optimal distribution for the components of Δ_k is symmetric Bernoulli, as suggested in Section III. This simple distribution has also proven effective in many finite-sample practical and simulation examples.[3] (Note that it is arbitrary as to whether to assume the magnitude of c or the magnitude of each element in Δ_k is unity since $c_k\Delta_k$ always appear together. Reference [25] assumes $c = 1$ and varies the magnitude of the Δ_k elements while the opposite approach is adopted here.)

Some extensions to the basic SPSA algorithm above 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 [31–33]. Reference [33] is the most complete treatment. Reference [31] also reports on a gradient smoothing idea (analogous to "momentum" in the neural network literature where gradients are averaged across iterations) 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). Simple gradient averaging (where several SP gradient approximations are averaged at each iteration) may also sometimes be useful in speeding convergence, even at the expense of the additional loss function evaluations (see footnote 2). An implementation of SPSA for global minimization is discussed in Chin [5] (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 [23] and Fu and Hill [12] using a projection approach. The "blocking" ideas in second-order SPSA (algorithm discussed below) may also be useful in standard SPSA

[3]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 towards the optimization, it may be worth evaluating other possibilities in some applications. For example, Maeda and De Figueiredo [17] 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.

to speed convergence and enhance stability; here an iteration is blocked (i.e., no θ update) if the new loss function value is significantly worse than the current loss value. This requires an extra loss evaluation at each iteration for monitoring and requires a choice regarding the amount of increase in a loss evaluation that is tolerable before a step is blocked (we have used the current loss value plus $2 \times$ [estimated noise standard deviation] as a nominal threshold above which the step is blocked).

A one (loss)-measurement form of the SP gradient approximation is considered in Spall [29]; although it is shown in this reference that the standard two-measurement form discussed here 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 (such as adaptive control or adaptive tracking) where the underlying system dynamics may change too rapidly to get a credible gradient estimate with two successive measurements. This form also allows for a so-called "one run" simulation optimization (e.g., Arsham [2]) by emulating the gradient-based (e.g., IPA) approaches with an estimate of the gradient using only one execution of a simulation. The major difference, of course, is that the SPSA-based one-run method does not require the detailed information about the "inner workings" of the simulation that is required in the gradient-based approaches (but the gradient-based approaches are likely to be more efficient if the required information is available at "reasonable" computational and other cost). The use of SPSA in simulation-based optimization is considered in detail in Kleinman, et al. [15], where it is shown that the method of "common random numbers" (i.e., among other factors, use of the same random number seed in the two measurements for the "standard" SPSA gradient approximation) can increase the rate of convergence.

An "accelerated" form of SPSA is reported in Spall [30]. 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 for any dimension p (and one additional measurement is recommended as a check on algorithm behavior and potential iteration "blocking"). The algorithm is implemented with two simple parallel recursions: one for θ and one for the Hessian of $L(\theta)$. The recursion for θ is a stochastic analogue of the well-known Newton–Raphson algorithm of deterministic optimization (which has the highly desirable property of being "transform invariant," i.e., it works equally well regardless of the units and/or relative magnitudes of the elements in θ). The recursion for the Hessian matrix is simply a recursive calculation of the sample mean of per-iteration Hessian estimates that are formed using SP-type ideas.

V. CONCLUDING REMARKS

This paper has presented some practical guidance on the implementation of the SPSA algorithm for stochastic optimization. Although these guidelines can be effective, the serious user would be advised to also become familiar with some of the theory behind SPSA and to possibly modify the guidelines using the theory and available prior information about the problem together with numerical experimentation. This paper also mentioned some extensions to the basic SPSA form, including global and constrained optimization forms, a version using only one loss measurement (versus two measurements), and a form representing a stochastic Newton–Raphson analogue.

APPENDIX. MATLAB® CODE

Fig. 1 presents MATLAB® code for performing n iterations of the standard (two loss measurement, first-order) SPSA algorithm in Section III. Algorithm initialization is not shown here since that can be handled in many ways (e.g., read from another file, direct inclusion in the program, user input during execution, etc.). 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
```

Fig. 1. Sample MATLAB® Code for SPSA. (Initialization for program variables theta, n, p, a, A, c, alpha, and gamma not shown.)

REFERENCES

[1] Alessandri, A., and Parisini, T. (1997)
Nonlinear modelling of complex large-scale plants using neural networks and stochastic approximation.
IEEE Transactions on Systems, Man, and Cybernetics—A, **27** (1997), 750–757.

[2] Arsham, H. (1998)
Algorithms for sensitivity information in discrete event systems simulation.
Journal of Simulation: Practice and Theory, **6** (1998), 1–22.

[3] Brennan, R. W., and Rogers, P. (1995)
Stochastic optimization applied to a manufacturing system operation problem.
In C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman (Eds.), *Proceedings of the Winter Simulation Conference*, 1995, 857–864.

[4] Cauwenberghs, G. (1994)
Analog VLSI autonomous systems for learning and optimization.
Ph.D. dissertation, California Institute of Technology, Pasadena, 1994.

[5] Chin, D. C. (1994)
A more efficient global optimization algorithm based on Styblinski and Tang.
Neural Networks, **7** (1994), 573–574.

[6] Chin, D. C. (1997)
Comparative study of stochastic algorithms for system optimization based on gradient approximations.
IEEE Transactions on Systems, Man, and Cybernetics, **27** (1997), 244–249.

[7] Chin, D. C., and Smith, R. H. (1994)
A traffic simulation for Mid-Manhattan with model-free adaptive signal control.
In *Proceedings of the Summer Computer Simulation Conference*, 1994, 296–301.

[8] Dennis, J. E., and Schnabel, R. B. (1989)
A view of unconstrained optimization.
In G. L. Nemhauser, A. H. G. Kan Rinnooy, and M. J. Todd (Eds.), *Optimization*, Handbooks in OR & MS, vol. 1.
New York: Elsevier, 1989, ch. 1, 1–72.

[9] Dippon, J., and Renz, J. (1997)
Weighted means in stochastic approximation of minima.
SIAM Journal of Control and Optimization, **35** (1997), 1811–1827.

[10] Fabian, V. (1971)
Stochastic approximation.
In J. J. Rustigi (Ed.), *Optimizing Methods in Statistics*.
New York: Academic Press, 1971, 439–470.

[11] Fu, M. C. (1994)
Optimization via simulation: A review.
Annals of Operations Research, **53** (1994), 199–247.

[12] Fu, M. C., and Hill, S. D. (1997)
Optimization of discrete event systems via simultaneous perturbation stochastic approximation.
Transactions of the Institute of Industrial Engineers, **29** (1997), 233–243.

[13] Hill, S. D., and Fu, M. C. (1995)
Transfer optimization via simultaneous perturbation stochastic approximation.
In *Proceedings of the Winter Simulation Conference*, 1995, 242–249.

[14] Kiefer, J., and Wolfowitz, J. (1952)
Stochastic estimation of a regression function.
Annals of Mathematical Statistics, **23** (1952), 462–466.

[15] Kleinman, N. L., Spall, J. C., and Naiman, D. Q. (1997)
Simulation-based optimization with stochastic approximation using common random numbers.
Management Science, submitted for publication.

[16] Maeda, Y., Hirano, H., and Kanata, Y. (1995)
A learning rule of neural networks via simultaneous perturbation and its hardware implementation.
Neural Networks, **8** (1995), 251–259.

[17] Maeda, Y., and De Figueiredo, R. J. P. (1997)
Learning rules for neuro-controller via simultaneous perturbation.
IEEE Transactions on Neural Networks, **8** (1997), 1119–1130.

[18] Nechyba, M. C., and Xu, Y. (1997)
Human-control strategy: Abstraction, verification, and replication.
IEEE Control Systems Magazine, **17**, 5 (1997), 48–61.

[19] Nelder, J. A., and Mead, R. (1965)
A simplex method for function minimization.
Computer Journal, **7** (1965), 308–313.

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

[21] Rezayat, F. (1995)
On the use of an SPSA-based model-free controller in quality improvement.
Automatica, **31** (1995), 913–915.

[22] Ruppert, D. (1983)
Kiefer–Wolfowitz procedure.
In S. Kotz and N. L. Johnson (Eds.), *Encyclopedia of Statistical Science*, Vol. 4.
New York: Wiley, 1983, 379–381.

[23] Sadegh, P. (1997)
Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation.
Automatica, **33** (1997), 889–892.

[24] Sadegh, P., and Spall, J. C. (1998)
Optimal sensor configuration for complex systems.
Journal of the American Statistical Association, submitted for publication (condensed version in *Proc. Am. Control Conf.*, 1998).

[25] Sadegh, P., and Spall, J. C. (1998)
Optimal random perturbations for multivariate stochastic approximation using a simultaneous perturbation gradient approximation.
IEEE Transactions on Automatic Control, **43** (1998), in press.

[26] Spall, J. C. (1987)
A stochastic approximation technique for generating maximum likelihood parameter estimates.
In *Proceedings of the American Control Conference*, 1987, 1161–1167.

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

[28] Spall, J. C. (1994)
Developments in stochastic optimization algorithms with gradient approximations based on function measurements.
In J. D. Tew, M. S. Manivannan, D. A. Sadowski, and A. F. Seila (Eds.), *Proceedings of the Winter Simulation Conference*, 1994, 207–214.

[29] Spall, J. C. (1997)
A one-measurement form of simultaneous perturbation stochastic approximation.
Automatica, **33** (1997), 109–112.

[30] Spall, J. C. (1997)
Accelerated second-order stochastic optimization using only function measurements.
In *Proceedings of the 36th IEEE Conference on Decision and Control*, 1417–1424.

[31] Spall, J. C., and Cristion, J. A. (1994)
Nonlinear adaptive control using neural networks: Estimation based on a smoothed form of simultaneous perturbation gradient approximation.
Statistica Sinica, **4** (1994), 1–27.

[32] Spall, J. C., and Cristion, J. A. (1997)
A neural network controller for systems with unmodeled dynamics with applications to wastewater treatment.
IEEE Transactions on Systems, Man, and Cybernetics, **27** (1997), 369–375.

[33] Spall, J. C., and Cristion, J. A. (1998)
Model-free control of nonlinear stochastic systems with discrete-time measurements.
IEEE Transactions on Automatic Control, **43** (1998), in press for issue no. 8.

[34] Wang, I.-J., and Chong, E. K. P. (1996)
A deterministic analysis of simultaneous perturbation stochastic approximation.
In *Proceedings of the 30th Conference on Information Sciences and Systems*, 1996, 918–922.

[35] White, H. (1989)
Some asymptotic results for learning in single hidden-layer feedforward network models.
Journal of the American Statistical Association, **84** (1989), 1003–1013.

James Spall (S'82—M'83—SM'90) joined The Johns Hopkins University, Applied Physics Laboratory in 1983 and was appointed to the Principal Professional Staff in 1991. He also teaches in The Johns Hopkins School of Engineering.

Dr. Spall has published many articles in the areas of statistics and control and holds two U.S. patents. For the year 1990, he received the Hart Prize as principal investigator of the most outstanding Independent Research and Development project at JHU/APL. He is an Associate Editor for the *IEEE Transactions on Automatic Control*, a Contributing Editor for the *Current Index to Statistics*, and he served as Editor and coauthor for the book *Bayesian Analysis of Time Series and Dynamic Models*. He is a member of the American Statistical Association, and a fellow of the engineering honor society Tau Beta Pi.