Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. James C. Spall

ISBN: 0-471-33052-3

CHAPTER 6

STOCHASTIC APPROXIMATION AND THE FINITE-DIFFERENCE METHOD

This and the next chapter continue in the spirit of Chapter 5 in focusing on the loss minimization problem with stochastic approximation (SA) (versus general root-finding as in Chapter 4). The approaches in this and the next chapter, however, differ in a fundamental way from the approach in Chapter 5. Namely, it is assumed that the stochastic gradient ($\partial Q/\partial \boldsymbol{\theta}$) is *not* available. Rather, it is assumed that only (generally noisy) measurements of the loss function are available. This gradient-free approach has significant value in many problems since the stochastic gradient is often difficult or impossible to obtain in practice.

Section 6.1 introduces some of the fundamental issues in gradient-free optimization and Section 6.2 discusses several problem areas where a gradient-free method is more appropriate than the gradient-based methods discussed in Chapter 5. Section 6.3 describes a core gradient-free SA method, which uses a classical finite-difference (FD) approximation to the gradient. Sections 6.4 and 6.5 summarize some of the theory available on convergence and convergence rates for the FD-based SA algorithm. Section 6.6 discusses the recurring issue of choosing the coefficients for the algorithm gain sequences. Section 6.7 presents some numerical results. Section 6.8 summarizes some extensions and provides a bridge to Chapter 7 on simultaneous perturbation SA. Section 6.9 offers some final remarks.

6.1 INTRODUCTION AND CONTRAST OF GRADIENT-BASED AND GRADIENT-FREE ALGORITHMS

As discussed in Chapters 4 and 5, stochastic approximation is a powerful tool for root-finding and optimization when only noisy measurements are available to solve the problem. Let us continue with the familiar problem of minimizing a differentiable loss function $L = L(\boldsymbol{\theta})$ via finding a root to $\partial L/\partial \boldsymbol{\theta} = \mathbf{0}$. Chapter 5 discussed the application of the root-finding (Robbins–Monro) SA methods (Chapter 4) to the minimization problem. This led to a stochastic gradient algorithm requiring knowledge of $\boldsymbol{Y}(\boldsymbol{\theta}) = \partial Q/\partial \boldsymbol{\theta}$ for Q in the representation $L(\boldsymbol{\theta}) = E[Q(\boldsymbol{\theta}, \boldsymbol{V})]$, where $\boldsymbol{V}$ corresponds to the cumulative randomness in the problem. Here $\partial Q/\partial \boldsymbol{\theta}$ represents a *direct* noisy measurement of the (unknown) true gradient $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L/\partial \boldsymbol{\theta}$.

There are, however, a large number of problems where the direct measurement, $\partial Q/\partial \boldsymbol{\theta}$, is difficult or impossible to obtain. For this reason, there is considerable interest in SA algorithms that do not depend on direct gradient measurements. Rather, these algorithms are based on an *approximation* to the gradient formed from (generally noisy) measurements of L. This interest has been motivated, for example, by problems in the adaptive control and statistical identification of complex systems, the optimization of processes by Monte Carlo simulations, the training of recurrent neural networks, the design of complex queuing and discrete-event systems, and the recovery of images from noisy sensor data. To contrast with the stochastic gradient algorithms of Chapter 5, the methods of this chapter are *gradient-free*. This use of "gradient-free" is associated with the *implementation* of the algorithms. It does not refer to conditions imposed to guarantee convergence, where the gradient $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L/\partial \boldsymbol{\theta}$ is assumed to exist.

The focus in this chapter is the SA algorithm based on the oldest method for gradient approximation—the finite-difference (FD) approximation (e.g., Dennis and Schnabel, 1989). The FD approximation relies on small one-at-a-time changes to each of the individual elements of $\boldsymbol{\theta}$. After each change, the (possibly noisy) value of L is measured. When measurements of L have been collected for perturbations in each of the elements of $\boldsymbol{\theta}$, the gradient approximation may be formed. The FD approach is motivated directly from the definition of the gradient as a collection of derivatives for each of the components in $\boldsymbol{\theta}$, holding all other components fixed. Unfortunately, the method can be very costly if the dimension p is high since one must collect at least one L measurement for each of the elements in $\boldsymbol{\theta}$. This cost motivates the Monte Carlo-based approaches of Section 6.8 and Chapter 7. Nevertheless, the FD method is fundamental in both stochastic and deterministic optimization.

Building on the seminal Robbins and Monro (1951) paper, an SA algorithm based on the FD gradient approximation was introduced for scalar θ in Kiefer and Wolfowitz (1952) and multivariate $\boldsymbol{\theta}$ in Blum (1954b). The FD-based SA (FDSA) algorithm is the oldest SA method using gradient approximations built from only loss measurements. Because of its relative ease of use, FDSA is much more widely used in practice than the stochastic gradient-based methods in at least one important area—simulation-based optimization (Fu and Hu, 1997, p. 5). It is probably more widely used in other areas as well, although this author can offer no proof of that.

Although they require only loss function measurements, gradient-free SA algorithms such as FDSA exhibit convergence properties similar to the properties of the stochastic gradient method of Chapter 5. The indirect connection to the gradient usually enhances the convergence when there is not a great danger of converging to an unacceptable local minimum. That is, basic FDSA—as with most other SA algorithms—is fundamentally a local optimizer. (Section 8.4 discusses global versions of SA.)

Recall that the random search methods of Chapter 2 also use only loss measurements (no gradients). Further, the random search methods may be *global* optimizers under appropriate conditions. However, one of the major restrictions in both the convergence theory for random search (such as Theorems 2.1 and 2.2) and in many practical implementations is the assumption of perfect *noise-free* loss measurements. This contrasts with FDSA, which is fundamentally based on noisy loss measurements (although a special case is noise-free measurements).

Direct measurements of the gradient $\partial Q/\partial \boldsymbol{\theta}$ do not typically arise naturally in the course of operating or simulating a system. Hence, one must have detailed knowledge of the underlying system input–output relationships in order to calculate the $\partial Q/\partial \boldsymbol{\theta}$ from basic output measurements such as the z_k in the nonlinear settings of Section 5.1. In contrast, the FDSA approach requires 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.

Because of the fundamentally different information needed in implementing the stochastic gradient-based and gradient-free algorithms, it is difficult to construct meaningful methods of comparison. As a general rule, however, the stochastic gradient algorithms converge faster *when speed is measured in number of iterations*. This is not surprising given the additional information required for the gradient-based algorithms. In particular, based on asymptotic distribution theory, one can determine the optimal rate of convergence of the iterate $\hat{\boldsymbol{\theta}}_k$ to an optimum $\boldsymbol{\theta}^*$. In a stochastic sense and for a large number of iterations k, this rate is proportional to $1/\sqrt{k}$ for the stochastic gradient (i.e., root-finding) algorithms (see Section 4.4) and proportional to the more slowly decaying $1/k^{1/3}$ for the algorithms based on gradient approximations (Fabian, 1971). (Exceptions to this maximum rate of convergence for the gradient-free algorithms are discussed in Fabian, 1971; Glasserman and Yao, 1992; L'Ecuyer and Yin, 1998; and Kleinman et al, 1999, where special cases are presented that achieve a rate arbitrarily close, or equal, to $1/\sqrt{k}$. See also Chapter 14 and the discussion on common random numbers.)

In practice, many factors besides the asymptotic rate of convergence must be considered in determining which algorithm is most appropriate for a given circumstance. The stochastic gradient algorithms may be either infeasible (if no system model is available) or undependable (if a poor system model is used). Further, the total cost to achieve effective convergence depends not only on the number of iterations required, but also on the cost per iteration, which is typically greater in gradient-based algorithms. This cost may include greater computational burden, additional human effort required for determining and writing software for gradients, and experimental costs for model building such as labor, materials, and fuel. Finally, the rates of convergence are based on asymptotic theory, and may not represent practical convergence rates in finite-samples. As a general rule, however, if direct (stochastic) gradient information is

conveniently and reliably available, it is generally to one's advantage to use this information in the optimization process. This chapter focuses on the setting where such information is *not* readily available.

6.2 SOME MOTIVATING EXAMPLES FOR GRADIENT-FREE STOCHASTIC APPROXIMATION

This section summarizes several problems where the stochastic gradient $\partial Q/\partial \boldsymbol{\theta}$ is difficult or impossible to obtain, motivating gradient-free SA methods such as FDSA or the simultaneous perturbation SA algorithm of the next chapter. The gradient-free methods use only (generally noisy) measurements of L. Three examples are given below, one in generic parameter estimation, one in feedback control, and one in simulation-based optimization.

Example 6.1—Generic parameter estimation for complex loss functions. Many general problems of statistical parameter estimation involve a complex relationship between $\boldsymbol{\theta}$ and $L(\boldsymbol{\theta})$, or, in the stochastic context, between $\boldsymbol{\theta}$ and $Q(\boldsymbol{\theta}, V)$. In such problems, difficulties in the calculation of the exact gradient $\boldsymbol{g}(\boldsymbol{\theta})$ or stochastic gradient $\partial Q/\partial \boldsymbol{\theta}$ may arise in several ways:

- It may be too costly in time and/or money to carry out the calculations required to obtain the gradient.
- There is the potential for human error in the derivations when it is possible—at least in principle—to carry out the calculations.
- For complex calculations, there is the possibility of software coding errors in implementing the algorithm even if the gradient derivation is correct.
- Although powerful for many low-dimensional problems of moderate difficulty, symbolic software packages such as MAPLE or MATHEMATICA have difficulty in handling the complex multivariate calculations of many serious gradient derivations.
- When trying to differentiate a function represented in software (e.g., taking the stochastic gradient of the output of a Monte Carlo simulation—see the example below), so-called *automatic differentiation* methods provide a means for calculating gradients (e.g., Griewank and Corliss, 1991). However, these methods require extensive knowledge of the "inner workings" of the software. While automatic differentiation provides a systematic means for gradient calculation, the term *automatic* is somewhat of a misnomer.

There are countless examples in the literature and/or in practice where it is difficult to derive the gradient. The classical method for avoiding unwieldy gradient calculations in optimization is to use the FD approximation to the gradient as mentioned in Section 6.1 (and discussed in detail in Section 6.3). The FD approximation can be used in approximating either the deterministic gradient $\boldsymbol{g}(\boldsymbol{\theta})$ or the stochastic gradient $\partial Q/\partial \boldsymbol{\theta}$ as appropriate. ❑

Example 6.2—Model-free feedback control system. The problem of parameter estimation above is one where it is *difficult* to obtain the gradient, but where—with enough care, patience, and resources—it *may* be possible to carry out the derivation. In contrast, we now present an example where it is *inherently impossible* to obtain the gradient.

Consider building a mathematical function that allows one to control and regulate a system when there is no mathematical model representing the open-loop dynamics of the system. The control function takes information about the current and past state of the system and produces the control signals to govern future behavior of the system. For example, in an urban vehicle traffic system with real-time control, the control function takes measurements of the traffic patterns throughout the network and produces the control signal governing the amount of green, yellow, and red time allotted to the intersections in the network.

The setup here differs from the customary control framework in that no attempt is made to model the system itself (the open-loop dynamics) when constructing the control function. Various forms of this model-free control approach are considered in Saridis (1977, pp. 375–376), Bayard (1991), and Spall and Cristion (1998). Two examples of available applications include Spall and Chin (1997) (vehicle traffic control) and Vorontsov et al. (2000) (adaptive optics). Applications in many other areas, including human–machine interface control, wastewater treatment, macroeconomic policy making, and chemical process control are given at *www.jhuapl.edu/SPSA* (also available through the book's Web site).

Let us consider the following special case for motivation. Suppose that $\boldsymbol{x}_k$ represents a state vector at time k; this vector is a summary of the relevant characteristics of the system. Let the process be governed by the following discrete-time recursion:

$$\boldsymbol{x}_{k+1} = \phi(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)\,, \tag{6.1}$$

where $\phi(\cdot)$ is an *unknown* input–output relationship (the open-loop system), $\boldsymbol{u}_k$ is the control signal, and $\boldsymbol{w}_k$ is random noise. One example of an automatic controller $\boldsymbol{u}_k$ is some function mapping that takes information about the current state and produces a signal to help the next state get close to its target vector. There exists some unknown best function for the controller depending on the unknown $\phi(\cdot)$. The aim here is to construct a control function *without* building a separate model for the unknown process $\phi(\cdot)$. Hence, in the traffic control example, no model is built for the traffic flow and queues (which, given the highly nonlinear and uncertain aspects of human behavior, is a virtually hopeless task in complex multiple-intersection networks).

The goal in the specific problem here is to have the state be close to a desired vector $\boldsymbol{d}_k$. In particular, one might aim to build a controller that minimizes the time-varying loss function

$$L_k(\boldsymbol{\theta}) = E[Q_k(\boldsymbol{\theta}, V_k)] = E\left(\tfrac{1}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{d}_{k+1}\|^2\right) \tag{6.2}$$

at each k, where V_k represents the cumulative randomness in the state $\boldsymbol{x}_{k+1}$ (e.g., that due to the cumulative effects of the noises $\boldsymbol{w}_0$, $\boldsymbol{w}_1$,..., $\boldsymbol{w}_k$ plus any other external random inputs). The need for the time index k on L_k and Q_k might be due, among other things, to the time variation in the target vector $\boldsymbol{d}_{k+1}$ (the dependence of the right-hand side of (6.2) on $\boldsymbol{\theta}$ is discussed below). Clearly, the dimensions of the vectors $\boldsymbol{x}_k$ and $\boldsymbol{d}_k$ must be identical.

Because the system relationship $\phi(\cdot)$ is unknown, it is not possible to know the analytical form of the true optimal control function. Therefore, we introduce an *approximation*

$$\boldsymbol{u}_k = \mu_k(\boldsymbol{x}_k, \boldsymbol{d}_{k+1} | \boldsymbol{\theta}), \tag{6.3}$$

where $\mu_k(\cdot | \boldsymbol{\theta})$ is a user-specified function approximator that depends on some parameters $\boldsymbol{\theta}$. (This building of a control function *without* the building of a separate model for the system $\phi(\cdot)$ is in the spirit of the principle of parsimony—Chapter 13—which favors a minimalist approach to modeling necessary to carry out the task at hand.)

As discussed in Section 5.2, one important class of "universal" function approximators is artificial neural networks. If a NN is used to represent the controller in (6.3), then the input to the NN at time k will be $\boldsymbol{x}_k$ and $\boldsymbol{d}_{k+1}$, the output will be the control signal, and $\boldsymbol{\theta}$ will represent the connection weights to be estimated. Note that $\boldsymbol{\theta}$ enters the right-hand side of (6.2) via the dependence of $\boldsymbol{x}_{k+1}$ on $\boldsymbol{u}_k$ and the dependence of $\boldsymbol{u}_k$ on $\boldsymbol{\theta}$. Figure 6.1 depicts a process associated with the setting of (6.1)–(6.3). Note the absence of any model for the system in the overall feedback process (hence the label *model free*). The details of the parameter estimation method are not shown in Figure 6.1, as they may vary depending on the nature of the system being controlled.

Let us now show why a gradient-based approach does not apply in the model-free control setting. For convenience, suppose that x_k and u_k are both scalars (but see Exercise 6.1). Returning to (6.2) and assuming that the necessary interchange of derivative and integral applies (as discussed in Section 5.1),

$$\frac{\partial Q_k}{\partial \boldsymbol{\theta}} = \frac{1}{2}\frac{\partial (x_{k+1} - d_{k+1})^2}{\partial \boldsymbol{\theta}} = (x_{k+1} - d_{k+1})\frac{\partial x_{k+1}}{\partial \boldsymbol{\theta}} = (x_{k+1} - d_{k+1})\frac{\partial \phi}{\partial u_k}\frac{\partial u_k}{\partial \boldsymbol{\theta}}. \tag{6.4}$$

The gradient $\partial\phi/\partial u_k$ is, however, unavailable since $\phi(\cdot)$ represents the *unknown* dynamics of the process. This breaks the chain in the chain rule on the right-hand side of (6.4). Hence, the overall stochastic gradient $\partial Q_k/\partial \boldsymbol{\theta}$ is unavailable even though $\partial u_k/\partial \boldsymbol{\theta}$ and $x_{k+1} - d_{k+1}$ are typically available (e.g., for a feedforward NN, $\partial u_k/\partial \boldsymbol{\theta}$ is readily available as part of the backpropagation calculations). So, as long as the dynamics are unknown, (6.4) implies that the gradient $\partial Q_k/\partial \boldsymbol{\theta}$ is unavailable. ❑

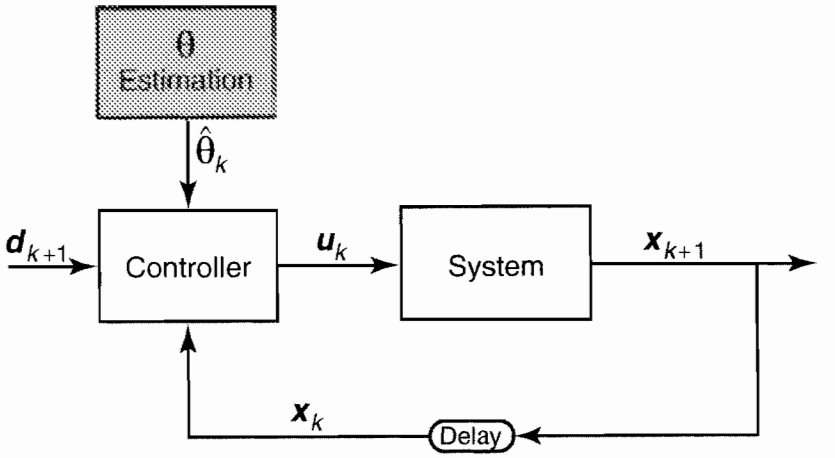

Figure 6.1. Model-free control setup with u_k representing the control input.

Example 6.3—Simulation-based optimization. Many practical systems are too complex to be analyzed via traditional analytical methods. Computer-based simulations are popular methods for investigating such systems. Suppose that one is going to use a simulation to optimize some real system. For example, in a public health scenario, there may be interest in determining some "best" combination of strategies related to vaccination, medical treatment, and quarantining, to stave off an epidemic. Here, $\boldsymbol{\theta}$ represents various terms associated with the available strategies. The simulation output represents some measure of public health (e.g., fraction of population that is ill), reflecting the value of $\boldsymbol{\theta}$ and the inherent randomness of the spread of disease.

For general problems, optimization of a *physical system* may be carried out by varying $\boldsymbol{\theta}$ in some intelligent way, running a *credible simulation* of the system, evaluating the outcome, and then repeating these steps until the "best" value of $\boldsymbol{\theta}$ has been obtained. In the notation of Chapter 5, $Q(\boldsymbol{\theta}, V)$ represents the simulation output and V represents the amalgamation of the Monte Carlo (pseudo) random effects in the simulation. The interest is in minimizing $L(\boldsymbol{\theta}) = E[Q(\boldsymbol{\theta}, V)]$, representing the mean outcome for a particular $\boldsymbol{\theta}$. So, each simulation output represents a noisy measurement of $L(\boldsymbol{\theta})$ (i.e., $y(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}, V)$).

For the very reason that a simulation is being used (versus analytical analysis), the functional relationship between $\boldsymbol{\theta}$ and $Q(\boldsymbol{\theta}, V)$ is likely to be very complex. Hence, it is unlikely that $\partial Q/\partial \boldsymbol{\theta}$ is available; knowledge of the "inner workings" of the simulation is required to obtain $\partial Q/\partial \boldsymbol{\theta}$. Rather, one may simply specify $\boldsymbol{\theta}$ and run the simulation to produce an output $Q(\boldsymbol{\theta}, V)$. This is an example of gradient-free optimization based on noisy measurements of the loss function. Chapters 14 and 15 consider simulation-based optimization in detail. ❑

6.3 FINITE-DIFFERENCE ALGORITHM

The above discussion indicates that many problems do not allow for the computation of $\partial Q/\partial\boldsymbol{\theta}$, as needed in the stochastic gradient form of the root-finding (Robbins–Monro) algorithm. This section introduces an alternative to the stochastic gradient algorithm for the case where only the measurements $y(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta})$ are available at various values of $\boldsymbol{\theta}$. In fact, the algorithm has the even weaker requirement of only requiring measurements of the *difference* of two values of the loss function, as opposed to measuring the loss functions themselves.

The recursive procedure here is in the general SA form

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k), \tag{6.5}$$

where $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is the estimate of the gradient $\partial L/\partial\boldsymbol{\theta}$ at the iterate $\hat{\boldsymbol{\theta}}_k$ based on measurements of the loss function. Hence, (6.5) is analogous to the stochastic gradient algorithm, with the gradient estimate $\hat{\boldsymbol{g}}_k(\boldsymbol{\theta})$ replacing the direct gradient measurement $\boldsymbol{Y}(\boldsymbol{\theta}) = \partial Q/\partial\boldsymbol{\theta}$ at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_k$. The gain $a_k > 0$ here acts in a way similar to its role in the stochastic gradient form. Under appropriate conditions, the iteration in (6.5) converges to $\boldsymbol{\theta}^*$ in some stochastic sense (usually almost surely, a.s.). Typical convergence conditions are very similar to those in Section 4.3 here for the root-finding SA algorithm. Convergence is discussed in Section 6.4.

The essential part of (6.5) is the gradient approximation $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$. We discuss below the oldest and best-known means of forming the approximation—the FD method. Expression (6.5) with this approximation represents the FDSA algorithm. One-sided gradient approximations involve measurements $y(\hat{\boldsymbol{\theta}}_k)$ and $y(\hat{\boldsymbol{\theta}}_k + \text{perturbation})$, while two-sided approximations involve measurements of the form $y(\hat{\boldsymbol{\theta}}_k \pm \text{perturbation})$. The two-sided FD approximation for use with (6.5) is

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} \dfrac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_1) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_1)}{2c_k} \\ \vdots \\ \dfrac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_p) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_p)}{2c_k} \end{bmatrix}, \tag{6.6}$$

where $\boldsymbol{\xi}_i$ denotes a vector with a 1 in the ith place and 0's elsewhere and $c_k > 0$ defines the difference magnitude. The pair $\{a_k, c_k\}$ are the gains (or gain sequences) for the FDSA algorithm. An obvious analogue to (6.6) holds for the

one-sided FD approximation: The ith component of the gradient approximation is $[y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_i) - y(\hat{\boldsymbol{\theta}}_k)]/c_k$. The two-sided form in (6.6) is the obvious multivariate extension of the scalar two-sided form in Kiefer and Wolfowitz (1952). The initial multivariate method in Blum (1954b) used a one-sided approximation.

6.4 CONVERGENCE THEORY

6.4.1 Bias in Gradient Estimate

The convergence theory for the FDSA algorithm is similar to the convergence theory for the root-finding SA algorithm as presented in Section 4.3. Additional difficulties, however, arise due to a bias in $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ as an estimator of $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$ and the need to control the extra gain sequence c_k. Recall from Section 4.3 that the standard "statistics" conditions for convergence of the iterate (A.1–A.4) required unbiased estimates of $\boldsymbol{g}(\cdot)$ at all k. (At the expense of other complications, the "engineering" conditions in Section 4.3 [B.1–B.5] required only an asymptotically unbiased estimate of $\boldsymbol{g}(\cdot)$.)

Let us begin by analyzing the bias in $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ as an estimator of $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$. Let $L_i'(\boldsymbol{\theta})$, $L_{ii}''(\boldsymbol{\theta})$, and $L_{iii}'''(\boldsymbol{\theta})$ represent the first, second, and third derivatives of L with respect to the ith component of $\boldsymbol{\theta}$. Suppose that the $L_{iii}''''(\boldsymbol{\theta})$ are continuous for all i and $\boldsymbol{\theta} \in \mathbb{R}^p$. Then by a third-order form of Taylor's theorem (Appendix A), we have

$$L(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_i) = L(\hat{\boldsymbol{\theta}}_k) + c_k L_i'(\hat{\boldsymbol{\theta}}_k) + \tfrac{1}{2}c_k^2 L_{ii}''(\hat{\boldsymbol{\theta}}_k) + \tfrac{1}{6}c_k^3 L_{iii}'''(\overline{\boldsymbol{\theta}}_k^{(i+)}), \quad (6.7a)$$

$$L(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_i) = L(\hat{\boldsymbol{\theta}}_k) - c_k L_i'(\hat{\boldsymbol{\theta}}_k) + \tfrac{1}{2}c_k^2 L_{ii}''(\hat{\boldsymbol{\theta}}_k) - \tfrac{1}{6}c_k^3 L_{iii}'''(\overline{\boldsymbol{\theta}}_k^{(i-)}), \quad (6.7b)$$

where $\overline{\boldsymbol{\theta}}_k^{(i+)}$ and $\overline{\boldsymbol{\theta}}_k^{(i-)}$ denote points on the line segments between $\hat{\boldsymbol{\theta}}_k$ and $\hat{\boldsymbol{\theta}}_k \pm c_k\boldsymbol{\xi}_i$ (so $\overline{\boldsymbol{\theta}}_k^{(i+)} = \hat{\boldsymbol{\theta}}_k + \lambda c_k\boldsymbol{\xi}_i$ for some $0 \le \lambda \le 1$; analogous for $\overline{\boldsymbol{\theta}}_k^{(i-)}$ with a generally different λ). Let $\mathfrak{I}_k = \{\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_k\}$. Suppose that the difference of the noise terms at any i, $\varepsilon(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_i) - \varepsilon(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_i)$, has conditional (on $\mathfrak{I}_k$) mean zero.

Combining (6.7a, b), the ith component of the FD gradient estimate (6.6) satisfies

$$E\left[\hat{g}_{ki}(\hat{\boldsymbol{\theta}}_k)\middle|\Im_k\right] = E\left[\frac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_i) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_i)}{2c_k}\middle|\Im_k\right]$$

$$= \frac{L(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_i) - L(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_i)}{2c_k}$$

$$= L_i'(\hat{\boldsymbol{\theta}}_k) + \frac{1}{6}\frac{c_k^3 L_{iii}'''(\bar{\boldsymbol{\theta}}_k^{(i+)}) + c_k^3 L_{iii}'''(\bar{\boldsymbol{\theta}}_k^{(i-)})}{2c_k}$$

$$\equiv L_i'(\hat{\boldsymbol{\theta}}_k) + b_{ki}, \tag{6.8}$$[1]

where $L_i'(\cdot)$ is the ith term of $\boldsymbol{g}(\cdot)$ and b_{ki} is the ith term of the bias

$$\boldsymbol{b}_k \equiv E\left\{\left[\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) - \boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)\right]\middle|\Im_k\right\}.$$

Assuming that the $L'''(\bar{\boldsymbol{\theta}}_k^{(i\pm)})$ terms are bounded, (6.8) implies that $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ has a bias of $O(c_k^2)$ (i.e., each component of $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ has an $O(c_k^2)$ bias):

$$E\left[\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)\middle|\Im_k\right] = \boldsymbol{g}(\hat{\boldsymbol{\theta}}_k) + \boldsymbol{b}_k;\ \boldsymbol{b}_k = O(c_k^2) \text{ a.s.}$$

Expression (6.8) also indicates that $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is an unbiased estimator of the gradient (i.e., $O(c_k^2) = \boldsymbol{0}$) in the special case where L is a quadratic loss function since $L_{iii}'''(\boldsymbol{\theta}) = 0$ for all i and $\boldsymbol{\theta}$. (See Exercise 6.2 for a discussion of the bias in the one-sided FD approximation.)

6.4.2 Convergence

Given the analysis of the bias above, we are now in a position to present conditions for the formal (a.s.) convergence of the FDSA algorithm. As with the root-finding SA algorithm, there are a large number of sets of sufficient conditions in the literature (see, e.g., Fabian, 1971; Kushner and Yin, 1997, Sects. 5.3, 8.3, and 10.3). The two sets of conditions below (analogues of the "statistics" and "engineering" conditions in Section 4.3) are representative of those in the literature, but are not the most general.

The conditions below are a special case of those in Fabian (1971, Theorem 2.4) (see also Ruppert, 1991). This set is presented in a manner roughly corresponding to the "statistics" conditions A.1–A.4 in Section 4.3 for the root-finding algorithm (i.e., A.j' is an extension or modification of A.j, j = 1, 2, 3, and 4). For convenience, let $\varepsilon_k^{(i\pm)} = \varepsilon(\hat{\boldsymbol{\theta}}_k \pm c_k\boldsymbol{\xi}_i)$.

[1] Strictly, all conditional expectations only hold a.s. For convenience, we drop the a.s. qualifier.

A.1′ **(Gain sequences)** $a_k > 0$, $c_k > 0$, $a_k \to 0$, $c_k \to 0$, $\sum_{k=0}^{\infty} a_k = \infty$, $\sum_{k=0}^{\infty} a_k c_k < \infty$, and $\sum_{k=0}^{\infty} a_k^2 / c_k^2 < \infty$.

A.2′ **(Unique minimum)** There is a unique minimum $\boldsymbol{\theta}^*$ such that for every $\eta > 0$, $\inf_{\|\boldsymbol{\theta}-\boldsymbol{\theta}^*\| > \eta} \|\boldsymbol{g}(\boldsymbol{\theta})\| > 0$ and $\inf_{\|\boldsymbol{\theta}-\boldsymbol{\theta}^*\| > \eta} [L(\boldsymbol{\theta}) - L(\boldsymbol{\theta}^*)] > 0$ (the "inf" statements here represent the infimum—see Appendix A—with respect to all $\boldsymbol{\theta}$ such that $\|\boldsymbol{\theta}-\boldsymbol{\theta}^*\| > \eta$).

A.3′ **(Mean-zero and finite variance noise)** For all i and k, $E\left[\left(\varepsilon_k^{(i+)} - \varepsilon_k^{(i-)}\right) \middle| \Im_k\right] = 0$ a.s. and $E\left[\left(\varepsilon_k^{(i\pm)}\right)^2 \middle| \Im_k\right] \leq C$ a.s. for some $C > 0$ that is independent of k and $\boldsymbol{\theta}$.

A.4′ **(Bounded Hessian matrix)** The Hessian matrix $\boldsymbol{H}(\boldsymbol{\theta}) = \partial^2 L / \partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T$ exists and is uniformly bounded in norm for all $\boldsymbol{\theta} \in \mathbb{R}^p$ (i.e., all components of $\boldsymbol{H}(\boldsymbol{\theta})$ are uniformly bounded in magnitude).

As with the root-finding SA algorithm, condition A.1′ is the most relevant from the point of view of the user's input. The condition includes restrictions on c_k as well as on a_k. It is apparent that $c_k \to 0$ slower than a_k. Condition A.2′ states that $\boldsymbol{\theta}^*$ is a unique global minimum in the sense that $L(\boldsymbol{\theta})$ always increases away from $\boldsymbol{\theta}^*$. Further, this condition rules out any local minima of L. A.3′ is the standard mean-zero and bounded variance noise condition. Although A.4′ places a bound on the Hessian, there is no need to assume the existence of third derivatives (as used in the bias discussion above); the third derivative assumption is used in the alternative conditions below.

There are also FDSA analogues of the "engineering" conditions B.1–B.5 for the root-finding approach (Section 4.3). Recall that the engineering conditions rely on connections of the SA recursion to an underlying ordinary differential equation (ODE). The FDSA algorithm can be expressed as a root-finding algorithm with a nonzero-mean noise (as allowed in condition B.5 of Section 4.3). Following the pattern of conditions A.1′–A.4′ above, we now present conditions B.1′–B.5′, which have a direct correspondence to B.1–B.5.

B.1′ **(Gain sequences)** $a_k > 0$, $c_k > 0$, $a_k \to 0$, $c_k \to 0$, $\sum_{k=0}^{\infty} a_k = \infty$, and $\sum_{k=0}^{\infty} a_k^2 / c_k^2 < \infty$.

B.2′ **(Relationship to ODE)** Same as condition B.2 in Section 4.3.

B.3′ **(Iterate boundedness)** Same as condition B.3 in Section 4.3.

B.4′ **(Mean and bounded variance of measurement error)** Same as condition A.3′ above.

B.5′ **(Smoothness of L)** The third derivatives $L_{iii}'''(\boldsymbol{\theta})$ are continuous and uniformly bounded for all $i = 1, 2, \ldots, p$ and $\boldsymbol{\theta} \in \mathbb{R}^p$.

The conditions above have largely been discussed in Section 4.3 or as part of the discussion following A.1′–A.4′. Because B.1′ does not include the condition $\sum_{k=0}^{\infty} a_k c_k < \infty$ appearing in A.1′, there is additional flexibility in picking the gains. In particular, the condition $\sum_{k=0}^{\infty} a_k c_k < \infty$ requires that a_k and c_k decay faster than sometimes recommended for practical applications (e.g., Section 6.6 suggests that a_k and c_k decay at rates proportional to $1/(k+1)^{0.602}$ and $1/(k+1)^{0.101}$ respectively; these gains satisfy B.1′ but not A.1′). On the other hand, B.2′ and B.3′ will generally be more formidable to verify than their closest counterpart A.2′. B.5′ is included to ensure that $\boldsymbol{b}_k = O(c_k^2)$ a.s., along the lines of the arguments surrounding (6.8).

The convergence conditions above provide an abstract ideal. In practice, one will rarely be able to check all of conditions A.1′–A.4′ or B.1′–B.5′ due to a lack of knowledge about L. In fact, the conditions may not be verifiable for the very reason that one is using the gradient-free FDSA algorithm! Nonetheless, the conditions are important in identifying the types of problems for which there are guarantees of algorithm convergence. Also, conditions on L that may be formally unverifiable may be at least intuitively plausible, providing some sense that the algorithm is appropriate for the problem.

We now present a convergence theorem for FDSA, together with a proof of the theorem. Although we omit most proofs in this book, this proof is included because of its relative accessibility (by connecting to the conditions of Theorem 4.1 for root-finding SA) and to convey some of the ideas that support SA. A reader may skip or skim the proof without great harm to subsequent understanding of FDSA.

Theorem 6.1. Consider the unconstrained problem (i.e., $\Theta = \mathbb{R}^p$). Suppose that conditions A.1′–A.4′ or conditions B.1′–B.5′ hold. Further, suppose that $\boldsymbol{\theta}^*$ is a unique minimum of L (i.e., Θ^* is the singleton $\boldsymbol{\theta}^*$). Then, for FDSA according to (6.5) and (6.6), $\hat{\boldsymbol{\theta}}_k \to \boldsymbol{\theta}^*$ a.s. as $k \to \infty$.

Proof. The proof under conditions A.1′–A.4′ is in Fabian (1971, Theorem 2.4). Let us work with conditions B.1′–B.5′. It is sufficient to show that these conditions imply or are equivalent to B.1–B.5 in Section 4.3 since B.1–B.5 allow for a biased gradient estimate. First, B.1′ implies B.1 through the addition of the decaying sequence c_k. There is nothing to show regarding B.2′ and B.3′ as these conditions directly assume the validity of the ODE (and related) conditions B.2 and B.3.

Let us now show the validity of B.4, which is satisfied if $E\left[\left\|\sum_{k=0}^{\infty} a_k(\boldsymbol{e}_k - \boldsymbol{b}_k)\right\|^2\right] < \infty$, where $\boldsymbol{b}_k = E[\boldsymbol{e}_k(\hat{\boldsymbol{\theta}}_k)\,|\,\Im_k]$. For the FDSA setting, $\boldsymbol{e}_k \equiv \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) - \boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$ since, in the notation of Chapter 4, $\boldsymbol{Y}_k(\hat{\boldsymbol{\theta}}_k) = \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$. Then

$$\lim_{n\to\infty} E\left[\left\|\sum_{k=0}^{n} a_k(\boldsymbol{e}_k - \boldsymbol{b}_k)\right\|^2\right] = \lim_{n\to\infty} \sum_{k=0}^{n} a_k^2 E\left(\left\|\boldsymbol{e}_k - \boldsymbol{b}_k\right\|^2\right), \tag{6.9}$$

where the cross products disappear in producing the right-hand side of (6.9) because $E[(\boldsymbol{e}_i - \boldsymbol{b}_i)^T(\boldsymbol{e}_j - \boldsymbol{b}_j)] = E\{E[(\boldsymbol{e}_i - \boldsymbol{b}_i)^T(\boldsymbol{e}_j - \boldsymbol{b}_j)|\Im_j]\} = E\{(\boldsymbol{e}_i - \boldsymbol{b}_i)^T E[(\boldsymbol{e}_j - \boldsymbol{b}_j)|\Im_j]\} = 0$ for all $i < j$ (the last equality follows since $\boldsymbol{e}_i - \boldsymbol{b}_i$ is uniquely determined by $\hat{\boldsymbol{\theta}}_i$ and $\hat{\boldsymbol{\theta}}_{i+1}$, both of which are contained in $\Im_j$). Note that

$$\boldsymbol{e}_k - \boldsymbol{b}_k = \begin{bmatrix} \dfrac{\varepsilon_k^{(1+)} - \varepsilon_k^{(1-)}}{2c_k} \\ \vdots \\ \dfrac{\varepsilon_k^{(p+)} - \varepsilon_k^{(p-)}}{2c_k} \end{bmatrix}. \tag{6.10}$$

Given the bounded variance from B.4′, (6.9) and (6.10) imply that $E\left[\left\|\sum_{k=0}^{\infty} a_k(\boldsymbol{e}_k - \boldsymbol{b}_k)\right\|^2\right]$ is proportional to $\sum_{k=0}^{\infty} a_k^2/c_k^2$, which, by B.1′, is bounded. This shows that B.4 in Section 4.3 is true. Finally, B.5′ implies, via the arguments surrounding (6.8), that $\boldsymbol{b}_k$ is uniformly bounded and has bias $O(c_k^2)$ a.s. So the bias satisfies the conditions $\sup_{k\geq 0}\|\boldsymbol{b}_k\| < \infty$ a.s. and $\boldsymbol{b}_k \to \boldsymbol{0}$ a.s. in B.5. Hence, all of B.1–B.5 hold under B.1′–B.5′, implying the a.s. convergence of the FDSA algorithm. ❑

6.5 ASYMPTOTIC NORMALITY

As with the root-finding SA algorithm, the FDSA iterates (appropriately standardized) have an asymptotic normal distribution under appropriate conditions. The asymptotic normality applies under more specific choices of the gain sequences than the general guidelines provided with the conditions for Theorem 6.1 in Section 6.4. In particular,

$$a_k = \frac{a}{(k+1+A)^{\alpha}} \text{ and } c_k = \frac{c}{(k+1)^{\gamma}}, \tag{6.11}$$

where a, c, α, and γ are strictly positive and the stability constant $A \geq 0$ plays the same role here that it did in Section 4.4. The results below on the large-sample

distribution of $\hat{\boldsymbol{\theta}}_k$ provide the asymptotically optimal values for the coefficients in (6.11).

Sacks (1958) was the first to establish the asymptotic normality of FDSA. Later, the more general result of Fabian (1968) on the asymptotic distribution of SA iterates was used to establish the asymptotic normality of FDSA under broader conditions than those of Sacks (1958). A simplified proof is given in Fabian (1971) for the $\alpha = 1$ special case. Generalized forms of asymptotic distributions are available via the *weak convergence* ideas in Kushner and Yin (1997, Chaps. 7 and 8); weak convergence is beyond the scope of this book.

Although we do not present a formal theorem here, the basic idea is to begin with an assumption that the algorithm converges (as in Theorem 6.1) and then add several conditions to guarantee asymptotic normality. Among the added conditions are

$$\beta \equiv \alpha - 2\gamma > 0 \text{ and } 3\gamma - \alpha/2 \geq 0. \tag{6.12}$$

Then, for the FDSA algorithm in (6.5) and (6.6) with gains as in (6.11),

$$k^{\beta/2}(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*) \xrightarrow{\text{dist.}} N(\boldsymbol{\mu}_{\text{FD}}, \boldsymbol{\Sigma}_{\text{FD}}) \tag{6.13}$$

as $k \to \infty$, where $\boldsymbol{\mu}_{\text{FD}}$ is a mean vector that depends on the Hessian $\boldsymbol{H}(\boldsymbol{\theta}^*)$ and the third derivative $L'''(\boldsymbol{\theta}^*)$, $\boldsymbol{\Sigma}_{\text{FD}}$ is some covariance matrix that depends on $\boldsymbol{H}(\boldsymbol{\theta}^*)$, and both $\boldsymbol{\mu}_{\text{FD}}$ and $\boldsymbol{\Sigma}_{\text{FD}}$ depend on the coefficients a, α, c, and γ of a_k and c_k (A does not affect the asymptotic properties). While the forms for $\boldsymbol{\mu}_{\text{FD}}$ and $\boldsymbol{\Sigma}_{\text{FD}}$ are somewhat unwieldy (see the references cited above), they are useful for shedding insight into the efficiency of FDSA. Chapter 7 uses these forms together with corresponding forms for another gradient-free SA algorithm to draw some conclusions about relative efficiency.

The presence of a generally nonzero mean ($\boldsymbol{\mu}_{\text{FD}}$) in the limiting distribution of FDSA is an interesting distinction from the limiting distributions commonly seen in estimation theory (including, e.g., the limiting distribution of root-finding SA as shown in Section 4.4). From the proofs of (6.13) in the references cited above, it is evident that this bias in the limiting distribution is a consequence of the bias in the gradient estimate. Note that the presence of $\boldsymbol{\mu}_{\text{FD}} \neq \mathbf{0}$ does *not* necessarily imply that $\hat{\boldsymbol{\theta}}_k$ has an asymptotic bias as an estimator of $\boldsymbol{\theta}^*$. In particular, under the additional conditions needed for the mean of the asymptotic distribution to correspond to the mean of the random process itself (e.g., Laha and Rohatgi, 1979, pp. 138–141), (6.13) implies that for large k, $E(\hat{\boldsymbol{\theta}}_k)$ is approximately equal to $\boldsymbol{\theta}^* + k^{-\beta/2}\boldsymbol{\mu}_{\text{FD}}$. Hence, $E(\hat{\boldsymbol{\theta}}_k)$ has a limiting value of $\boldsymbol{\theta}^*$. (Note that the conditions for asymptotic normality alone—as for (6.13)—do *not* guarantee that $\lim_{k\to\infty} E[k^{\beta/2}(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*)] = \boldsymbol{\mu}_{\text{FD}}$; see Exercise 6.4.)

Akin to the discussion in Section 4.4 for the root-finding SA algorithm, expression (6.13) implies that the stochastic rate at which $\hat{\boldsymbol{\theta}}_k$ approaches $\boldsymbol{\theta}^*$ is proportional to $k^{-\beta/2}$ for large k. That is, $\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*$ must be decaying in a stochastic sense at a rate proportional to $k^{-\beta/2}$ to balance the increasing $k^{\beta/2}$ term on the left-hand side of (6.13), yielding a well-behaved random vector with the distribution $N(\boldsymbol{\mu}_{FD}, \boldsymbol{\Sigma}_{FD})$.

With the gain forms in (6.11), and under condition B.1′ (weaker than A.1′) for convergence of the iterate, we know that $\alpha > 1/2$ and $\gamma > 0$. The conditions in (6.12) put further constraints on α and γ, implying that

$$0.6 < \alpha \leq 1, \ 0.1 < \gamma < 1/2, \text{ and } \alpha - \gamma > 1/2 \tag{6.14}$$

(see Exercise 6.5). Note that the reverse implication does not hold: an arbitrary α and γ satisfying (6.14) may not satisfy the gain conditions. For example, $\alpha = 1$ and $\gamma = 0.15$ satisfy (6.14) but violate $3\gamma - \alpha/2 \geq 0$ in (6.12). Given B.1′ and (6.12), we find that β is maximized at $\alpha = 1$ and $\gamma = 1/6$, leading to a maximum rate of stochastic convergence of $\hat{\boldsymbol{\theta}}_k$ to $\boldsymbol{\theta}^*$ that is proportional to $k^{-\beta/2} = 1/k^{1/3}$ for large k. This contrasts with a maximum rate proportional to $1/\sqrt{k}$ in the stochastic gradient algorithms of root-finding type (see Section 4.4). Hence, the direct gradient information of the stochastic gradient implementation "buys" an increase in the maximum rate of convergence from $O(1/k^{1/3})$ to $O(1/\sqrt{k})$.

Glasserman and Yao (1992) and L'Ecuyer and Yin (1998) show that the maximum rate of convergence in FDSA can be the same as in root-finding SA (i.e., $O(1/\sqrt{k})$) in the special case of simulation-based optimization with common random numbers. This contrasts with the standard maximum rate of $O(1/k^{1/3})$. This issue is considered in detail in Chapter 14.

6.6 PRACTICAL SELECTION OF GAIN SEQUENCES

This section focuses on the selection of the coefficients a, A, c, α, and γ in the gains a_k and c_k appearing in (6.11). As with root-finding SA, the asymptotic analysis above may not point to the best choice of gains in practical *finite-sample* problems. It is often (but not always!) preferable in practice to have a slower decay rate than the asymptotically optimal $\alpha = 1$ and $\gamma = 1/6$. This provides more power to the algorithm through larger step sizes when k is large. Practical values of α and γ that are effectively as low as possible while satisfying B.1′ and (6.12) are 0.602 and 0.101, respectively.

In practical applications, the gains are usually chosen by trial and error on some small-scale (e.g., reduced number of iterations) version of the full problem. Obviously, one typically wants a minimal number of the overall "budget" of loss measurements to be devoted to the gain tuning process so that most of the effort in collecting loss measurements is directed to the optimization process per se.

It is sometimes possible to partially automate the gain selection process, as we now outline in a method referred to as the *semiautomatic* method for gain selection. The aim is to pick the coefficients a, A, c, α, and γ in the gains of (6.11). First, picking $\alpha = 0.602$ and $\gamma = 0.101$, as mentioned above, provide the generally more desirable slowly decaying gains. To cope with noise effects, it is effective to set c at a level approximately equal to the standard deviation of the measurement noise in $y(\boldsymbol{\theta})$ (see Exercise 6.9). This helps keep the p elements of $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ from getting excessively large in magnitude before a_k has decreased enough to compensate during the search process (the standard deviation can be estimated by collecting several $y(\boldsymbol{\theta})$ values at the initial guess $\hat{\boldsymbol{\theta}}_0$). If the standard deviation changes dramatically with $\boldsymbol{\theta}$, this approach might not be useful. In the case where one has perfect (noise-free) measurements of $L(\boldsymbol{\theta})$, then c should be chosen as some small positive number.

The values of a and A can be chosen together to ensure effective practical performance of the algorithm. As discussed in Section 4.4, a stability constant $A > 0$ allows for a more aggressive search (larger a) through avoiding instabilities in the early iterations. A larger a often enhances performance in the later iterations by producing a larger step size when the effect of A is negligible compared to the iteration count k. A rule of thumb is to choose A such that it is much less than the maximum number of iterations allowed or expected (e.g., take it to be 10 percent or less of the maximum number of expected/allowed iterations).

After determining A, one must choose the most important coefficient, a. In making this choice, first pick $a_{\text{temp},i}$, $i = 1, 2, \ldots, p$, such that $\left[a_{\text{temp},i}/(A+1)^{0.602}\right]\hat{g}_{0i}(\hat{\boldsymbol{\theta}}_0)$ is approximately equal to the desired change magnitude in the ith element of $\boldsymbol{\theta}$ in the early iterations, where $\hat{g}_{0i}(\hat{\boldsymbol{\theta}}_0)$ represents the i^{th} element of $\hat{\boldsymbol{g}}_0(\hat{\boldsymbol{\theta}}_0)$. To do this reliably in the face of the noise in the loss measurements may require several replications of $\hat{\boldsymbol{g}}_0(\hat{\boldsymbol{\theta}}_0)$; the typical magnitude for the ith element of the gradient estimate is then taken to be an average of $\left|\hat{g}_{0i}(\hat{\boldsymbol{\theta}}_0)\right|$ from several replications. Finally, $a = \min\{a_{\text{temp},1}, a_{\text{temp},2}, \ldots, a_{\text{temp},p}\}$, where the *minimum* is chosen to preserve the stability of the algorithm.[2] (At the expense of a more cumbersome implementation, it is also possible to use a diagonal *matrix* gain $\boldsymbol{a}_k$ with the ith diagonal element being the gain in (6.11) with numerator $a = a_{\text{temp},i}$.)

Note that this semiautomatic means of picking the gain coefficients is generally useful in getting *reasonable* values. In most applications, refinement of the values will enhance the performance of the algorithm. Unfortunately, this refinement is usually possible only after observing the search process or by

[2]This is an illustration of the *lack* of transform invariance of a first-order algorithm. Because (6.11) does not scale for each element of $\boldsymbol{\theta}$ separately, the gain is chosen conservatively so that no "wild" behavior is seen for any one element. A matrix gain—such as a second-order algorithm in Subsection 4.5.2—scales each element differently.

doing "trial runs" where the gains are tuned to provide better results. Usually (not always!), the greatest benefit follows by focusing this refinement on the single coefficient a, as the search tends to be more robust to changes in the other coefficients (A, c, α, and γ). We demonstrate the semiautomatic method of gain selection in the example below; the method is used in the studies in Section 6.7.

Example 6.4—Choosing gain coefficients using the semiautomatic method. Suppose that a maximum of 6000 loss measurements may be used during the search process for a $p = 3$ problem. Further, based on several measurements of the loss function at the initial condition, $y(\hat{\boldsymbol{\theta}}_0)$, it has been determined that the noise has an approximate standard deviation of 0.5 and that the standard deviation is not expected to change significantly as $\boldsymbol{\theta}$ changes. Then, together with $\alpha = 0.602$ and $\gamma = 0.101$, one chooses $c = 0.5$. Using a threshold of 10 percent of the maximum number of iterations implies that $A = 0.10 \times 6000/(2 \times 3) = 100$. Suppose further that it is felt that the elements of $\boldsymbol{\theta}$ should typically move by a magnitude 0.1 in the early iterations. Based on the c chosen above, one forms several $\hat{\boldsymbol{g}}_0(\hat{\boldsymbol{\theta}}_0)$, finding that the magnitudes of $\hat{g}_{0,1}(\hat{\boldsymbol{\theta}}_0)$, $\hat{g}_{0,2}(\hat{\boldsymbol{\theta}}_0)$, and $\hat{g}_{0,3}(\hat{\boldsymbol{\theta}}_0)$ are around 10, 20, and 10, respectively. This leads to $a_{\text{temp},1} = a_{\text{temp},3} = 0.16$ and $a_{\text{temp},2} = 0.08$ (i.e., for the first and third elements, $[0.16/(100+1)^{0.602}] \times 10 = 0.1$). Hence, the choice is $a = 0.08$. All five of the gain coefficients have now been chosen. It is likely that refinements—especially of a—will be helpful after one has observed the optimization process. ❑

6.7 SEVERAL FINITE-DIFFERENCE EXAMPLES

Let us present three examples of FDSA: one for a simple $p = 2$ problem and two for a more challenging $p = 10$ problem. In comparing the performance of FDSA with the random search methods of Chapter 2, these examples illustrate the benefits of the assumptions about L that are required in FDSA. In particular, while the no free lunch theorems (Subsection 1.2.2) indicate that all algorithms perform the same when averaging over all possible problems, we have prior information stating that L is differentiable and unimodal. This prior information provides enough structure for an algorithm such as FDSA (which uses the *existence* of the gradient) to have an apparent advantage over more general algorithms such as the random search algorithms of Chapter 2. (One must be cautious about the connections to the NFL theorems because of NFL's restriction to discrete problems; nevertheless, since the implementation of FDSA is on a digital machine, FDSA is strictly a discrete algorithm.) This advantage applies even if, as with FDSA, the *mechanics* of the algorithm do not depend explicitly on this prior information. In particular, both FDSA and random search use the same information—noisy or noise-free loss measurements—in the operations of the algorithm.

Example 6.5—Optimization in wastewater treatment problem (redux). Let us revisit Examples 2.6–2.8, which consider a $p = 2$ problem related to wastewater treatment. The aim is to run FDSA on this problem and compare results with the random search results in Example 2.7 using the same number of loss measurements n (so each run of FDSA is for $n/(2p) = n/4$ iterations). As in Examples 2.6–2.8, the random search method used here is algorithm B, described in Sections 2.2 and 2.3 for noise-free and noisy problems. The problem setup of Example 2.7 is used here, including measurement process $y(\boldsymbol{\theta})$, noise level, initial condition ($\hat{\boldsymbol{\theta}}_0$), priority of weighting on methane gas and water cleanliness, and definition of Θ ($= [0, 5] \times [0, 5]$). To avoid confusion in reviewing Examples 2.6–2.8, note that because of the current focus on optimization via finding a solution to $\partial L/\partial \boldsymbol{\theta} = \mathbf{0}$, the mathematical form of $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L/\partial \boldsymbol{\theta}$ here generally differs slightly from $\boldsymbol{g}(\boldsymbol{\theta})$ as used in Examples 2.6–2.8 (where $\boldsymbol{g}(\boldsymbol{\theta})$ is a generic root-finding function from which L is built).

If an iterate falls outside of Θ, each violating component of $\boldsymbol{\theta}$ is mapped to it nearest valid point in Θ. The subsequent FD gradient estimate is formed at the modified (valid) $\boldsymbol{\theta}$ value. For example, an iterate achieving the value $\boldsymbol{\theta} = [2.2, 6.2]^T$ via the update step in (6.5) is mapped to $\boldsymbol{\theta} = [2.2, 5.0]^T$; the next FD approximation is computed at $[2.2, 5.0]^T$. (The perturbed values $\hat{\boldsymbol{\theta}}_k \pm c_k \boldsymbol{\xi}_i$ for FDSA are allowed to go outside of Θ.)

Table 6.1 contrasts the mean values of $L(\hat{\boldsymbol{\theta}}_k) - L(\boldsymbol{\theta}^*)$ at the terminal iteration over 50 independent replications. The numbers in the table can be directly compared with the numbers in the table of Example 2.7. Below the means are 95 percent confidence intervals calculated from a t-distribution as in Appendix B. The first column of results is based on naïve gains $a_k = 1/(k+1)$ and $c_k = 1/(k+1)^{1/6}$, corresponding to the asymptotically optimal α and γ according to the discussion in Section 6.5. The second column uses gains of form (6.11) that have been partially tuned based on trial and error to adjust the numerator a in a_k to provide optimal results while maintaining the same $\alpha = 1$, $\gamma = 1/6$, $c = 1$, and $A = 0$ ($a_k = a/(k+1)$ with $a = 0.4$ for $n = 100$ loss measurements; $a = 1.5$ for $n = 2000$; $c_k = 1/(k+1)^{1/6}$ in both cases). Hence, the

Table 6.1. Sample means (50 replications) of $L(\hat{\boldsymbol{\theta}}_k) - L(\boldsymbol{\theta}^*)$ for FDSA under two a_k sequences; values contrast with $L(\hat{\boldsymbol{\theta}}_0) - L(\boldsymbol{\theta}^*) = 38.43$. Approximate 95 confidence intervals are also shown.

	FDSA with "naïve" gains	FDSA with partially tuned gains
$n = 100$ ($k = 25$ iterations)	0.11 [0.087, 0.140]	0.083 [0.057, 0.108]
$n = 2000$ ($k = 500$ iterations)	0.023 [0.017, 0.028]	0.021 [0.016, 0.026]

gain tuning did not involve the full complement of flexibility for the gains in (6.11).

Based on numerical experimentation with $n = 2000$, the asymptotically optimal $\alpha = 1$ and $\gamma = 1/6$ seemed to provide performance superior to that with the recommended finite-sample values of $\alpha = 0.602$ and $\gamma = 0.101$. This appears to be a consequence of asymptotic theory when using 2000 measurements to estimate only two parameters. (The larger-dimensional Example 6.6 shows a case where $\alpha = 0.602$ and $\gamma = 0.101$ provide superior performance.)

The four sample means in this table are significantly below the corresponding values in Example 2.7 based on random search algorithm B. For example, at $n = 2000$, the best value in Example 2.7 is 0.38, significantly greater than 0.021 or 0.023 as shown above. Further, the confidence intervals here and in Example 2.7 are not close to overlapping. This illustrates the benefit of the partial information about the gradient $\boldsymbol{g}(\boldsymbol{\theta})$ that is available from the FD approximation.

In comparing the numbers *within* Table 6.1, note that the confidence intervals for the naïve and tuned cases overlap for each n value. Although this may suggest nonsignificant differences between the naïve and tuned cases, one should dig a little further because the same random number seed is used to initialize all runs, providing a degree of shared randomness. The matched-pairs test (Appendix B) is the proper test in such a case. Based on the 50 terminal loss values, this t-test reveals a nonsignificant difference between the values for $n = 2000$ (two-sided P-value of 0.18) and a significant difference between the values for $n = 100$ (two-sided P-value of 0.009). ❑

The next two examples are for a larger $p = 10$ problem. The primary aims are to illustrate the role of "gain tuning" and its effect on the quality of the solution, to compare FDSA against random search algorithm B from Chapter 2, and to illustrate that the closeness of $L(\hat{\boldsymbol{\theta}}_k)$ and $L(\boldsymbol{\theta}^*)$ does not necessarily translate into $\hat{\boldsymbol{\theta}}_k$ being close to $\boldsymbol{\theta}^*$. The loss function considered here is a fourth-order (i.e., quartic) polynomial with significant variable interaction and highly skewed level surfaces. By *highly skewed*, we mean that a given change in one of the $\boldsymbol{\theta}$ elements may affect L in a way very different than the same change in another element of $\boldsymbol{\theta}$. The first of the two examples, Example 6.6, compares algorithms based on their performance relative to the true loss function. The second example, Example 6.7, compares algorithms based on the accuracy of the final $\boldsymbol{\theta}$ estimate.

Example 6.6—Skewed quartic loss function: comparison of loss values in FDSA and random search. Consider the loss function

$$L(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{B}^T \boldsymbol{B}\boldsymbol{\theta} + 0.1\sum_{i=1}^{p} (\boldsymbol{B}\boldsymbol{\theta})_i^3 + 0.01\sum_{i=1}^{p} (\boldsymbol{B}\boldsymbol{\theta})_i^4, \tag{6.15}$$

where $(\cdot)_i$ represents the ith component of the argument vector $\boldsymbol{B}\theta$, and $\boldsymbol{B}$ is such that $p\boldsymbol{B}$ is an upper triangular matrix of 1's (so elements below the diagonal are zero). Let $p = 10$. The minimum occurs at $\theta^* = 0$ with $L(\theta^*) = 0$; all runs are initialized at $\hat{\theta}_0 = [1, 1, \ldots, 1]^T$ (so $L(\hat{\theta}_0) = 4.178$). Unlike Example 6.5, the measurement noise ε is independent, identically distributed (i.i.d.); the distribution of the noise is $N(0, 1)$. All iterates are constrained to be in $\Theta = [-5, 5]^{10}$ (the 10-dimensional hypercube with minimum and maximum values of -5 and 5). The ratio of maximum to minimum eigenvalue of the Hessian $\boldsymbol{H}(\theta^*)$ is approximately 65, indicating the high degree of skewness in (6.15).

Using $n = 1000$ loss measurements for all algorithms, the aim here is to compare estimation performance of: (i) FDSA ((6.5) and (6.6)) using the semiautomatic gain selection method outlined in Section 6.6, (ii) FDSA using the gains determined by manual trial and error following the semiautomatic gain selection, and (iii) random search algorithm B from Sections 2.2 and 2.3. For FDSA, the gains are of the form (6.11), while for algorithm B, trial and error is used to choose the approximate best value of the perturbation distribution and the amount of averaging needed to cope with the noise (Section 2.3). For both random search and FDSA, a point θ lying outside of Θ is moved back to Θ on a component-wise basis in the same manner as Example 6.5 (so, in algorithm B, it is not necessary to generate extra perturbations $\boldsymbol{d}_k$, and associated loss measurements, to meet the constraints, as was mentioned in step 1 of the algorithm in Subsection 2.2.2).

The coefficient values used for the three algorithm implementations are shown in Exercises 6.10 and 6.11; for the manual tuning, a, A, α, and γ (but not c) are varied and experimentally tested, using the semiautomatic values as a starting point in the experiments. Note that both FDSA with manual tuning and algorithm B required a number of trial runs over *all* 1000 measurements. (In practice, it would be possible to use truncated runs in the hopes of getting good coefficient values for the later full-length runs.) FDSA-semiautomatic, on the other hand, required only a small number of trial loss measurements around $\hat{\theta}_0$. We used 100 trial measurements here, producing five gradient approximations to be averaged in determining the "typical" magnitudes of the gradient elements. This is a major advantage in practical applications, where it is not typically possible to collect the required *multiple sets* of the full complement of measurements (e.g., 1000 here) to tune for *one* later run using the same number of measurements.

Figure 6.2 summarizes the results. Each curve represents the sample mean of 50 independent replications. An individual replication of any one of the three algorithms has much more wiggle than the corresponding smoothed curve in the figure. As indicated in Exercise 6.11, algorithm B has 20 measurements being averaged for each loss evaluation. Hence, the 1000 measurements translates into $1000/(2\times10) = 50$ iterations for FDSA and 49 iterations for algorithm B (including the initial loss evaluation).

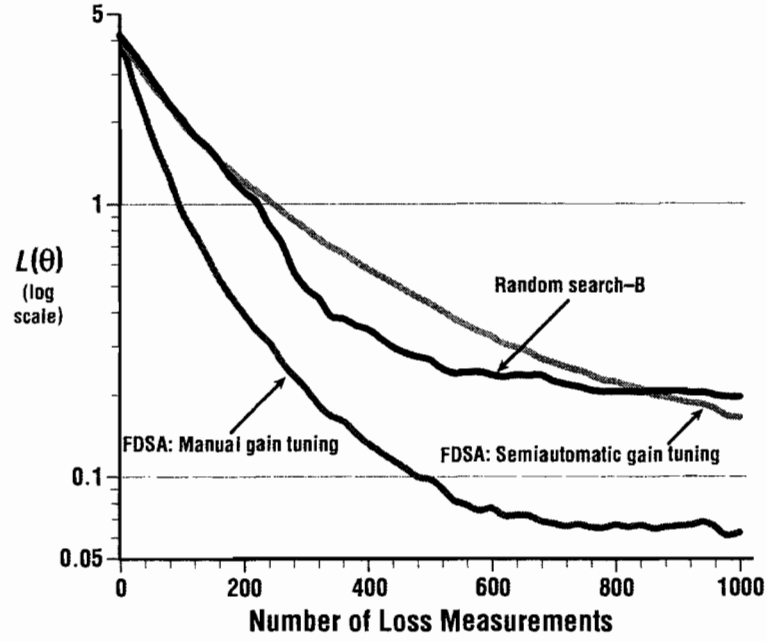

Figure 6.2. Comparison of two implementations of FDSA (with semiautomatic gains and tuned gains) and random search algorithm B. Each curve represents sample mean of 50 independent replications.

Figure 6.2 shows that all algorithm implementations produce an overall reduction in the loss function as the number of measurements approach 1000 (the true loss values $L(\hat{\boldsymbol{\theta}}_k)$ are used in constructing the figure; these values are not available to the algorithms, which use only noisy measurements $y(\boldsymbol{\theta})$ at the various values of $\boldsymbol{\theta}$). The curves illustrate that the FDSA algorithm with manual tuning performs (as expected) the best, but that the FDSA algorithm with semiautomatic tuning performs reasonably well. The manual tuning experiments reveal that $\alpha = 0.602$ and $\gamma = 0.101$ work better here than the asymptotically optimal $\alpha = 1$ and $\gamma = 1/6$, suggesting that 1000 measurements for $p = 10$ parameters is not yet asymptotic-like in its behavior for this problem (contrast with Example 6.5). Random search algorithm B outperforms the FDSA-semiautomatic algorithm for most of the search period, but the situation reverses after approximately 840 loss measurements. ❑

Example 6.7—Skewed quartic loss function: comparison of the accuracy of $\boldsymbol{\theta}$ estimates in FDSA and random search. Let us continue with the setting of Example 6.6. Interesting insight comes by looking at the accuracy of $\hat{\boldsymbol{\theta}}_k$ relative to $\boldsymbol{\theta}^*$, in contrast to Example 6.6 based on comparing $L(\hat{\boldsymbol{\theta}}_k)$ with $L(\boldsymbol{\theta}^*)$. Although, the loss function is formally the fundamental criterion of interest, it is often of interest in practice to be concerned with the distance from $\hat{\boldsymbol{\theta}}_k$ to $\boldsymbol{\theta}^*$

(i.e., $\|\hat{\theta}_k - \theta^*\|$). This is especially true in cases—such as maximum likelihood estimation—where the criterion L may not have an easy physical interpretation.

For each of the three algorithms considered in Figure 6.2, Table 6.2 shows the sample mean of the distances from $\hat{\theta}_k$ to θ^* for the 50 final estimates formed from the 1000 measurements. (Note: The table values are *not* the root-mean-squared values corresponding to the square root of the sample mean of the mean-squared errors.) The $\hat{\theta}_k$ values producing the sample means in Table 6.2 are identical to the terminal estimates generating Figure 6.2. The distances are normalized by $\|\hat{\theta}_0 - \theta^*\| = 3.162$. The approximate 95 percent confidence intervals are calculated in the usual manner from a t-distribution.

There are two notable points in Table 6.2. One is that the FDSA-semiautomatic algorithm provides the best performance, unlike the results of Figure 6.2 (note the nonoverlap in the confidence intervals). This confirms a point that is easy to forget: A "better" θ value with respect to L is not necessarily closer to θ^* than a value with a poorer L value. In fact, the table shows that with algorithm B, the final solutions are typically *farther* from θ^* than the initial condition, although the mean of the corresponding loss values is reduced by approximately 80 percent. Such a disparity between loss and distance measures is typical for highly skewed loss functions, where the θ sensitivities vary considerably by element. An element with little impact on L may be poorly estimated, barely affecting the loss value at the final estimate, but having a devastating effect in the distance from the final estimate to θ^*.

The other interesting point in Table 6.2 is that the magnitudes of the numbers are much larger than the corresponding numbers from Figure 6.2 when normalized by the initial loss value $L(\hat{\theta}_0) = 4.178$ (corresponding to the normalization by $\|\hat{\theta}_0 - \theta^*\|$ in Table 6.2). These normalized *loss* values are on the order of 10^{-2}. This relative behavior is typical of most optimization problems, reflecting the relative flatness of the loss surface in the area surrounding θ^*. ❑

Table 6.2. Comparison of FDSA and random search algorithm B in accuracy of θ. Algorithms use 1000 loss measurements per replication. Values shown are sample means over 50 replications with approximate 95 percent confidence intervals.

	FDSA: semiautomatic gains	FDSA: manually tuned gains	Random search B
Normalized error $\frac{\|\hat{\theta}_k - \theta^*\|}{\|\hat{\theta}_0 - \theta^*\|}$	0.427 [0.411, 0.443]	0.531 [0.502, 0.561]	1.285 [1.190, 1.378]

6.8 SOME EXTENSIONS AND ENHANCEMENTS TO THE FINITE-DIFFERENCE ALGORITHM

The sections above discussed the basic FDSA algorithm. Let us now summarize a few extensions of the basic method.

Fabian (1967, 1971) presents several methods for accelerating the convergence of FDSA-type algorithms, analogous to some of the methods in Subsection 4.5.2 for root-finding SA. The methods are based on taking additional measurements to explore the loss function surface in detail. One such method in Fabian (1971) is a stochastic analogue to second-order algorithms of the generic Newton–Raphson form. This algorithm uses $O(p^2)$ measurements $y(\cdot)$ per iteration for the gradient and Hessian estimation. The gradient is estimated in a standard way (e.g., (6.6)) and the Hessian is estimated using an analogous FD-based double-differencing scheme. Although this method is intuitively sensible, it demands many extra loss measurements if p is even moderately large and is likely to be numerically unstable with even a small level of noise. There appear to be no serious published applications or numerical evaluations of this idea. Further, a more efficient scheme for approximating the Hessian matrix is given in Section 7.8.

The iterate averaging approach of Subsection 4.5.3 may also be used with FDSA. Dippon and Renz (1997) theoretically analyze iterate averaging in the FDSA context. Unlike the root-finding setting, Dippon and Renz (1997) find that because of the biases in the underlying gradient approximations, iterate averaging does *not* generally yield asymptotically optimal performance. However, the difference between the iterate averaging solution and the true optimal solution based on the best (intractable) gain sequence a_k is *quantifiably* bounded. The true optimal gain sequence a_k depends on the unknown third derivatives of L evaluated at the unknown $\boldsymbol{\theta}^*$. This bound suggests that iterate averaging yields "good"—but not the best—asymptotic behavior for FDSA. Further, the practical finite-sample issues limiting the performance of iterate averaging, as discussed in Subsection 4.5.3, apply here as well.

Injected Monte Carlo randomness—as in Property B in Subsection 1.1.3—can be used in combination with FDSA-type methods. This is the essence of the simultaneous perturbation SA method of Chapter 7. Ermoliev (1969) was apparently the first to introduce such an idea via random directions SA (RDSA). The essential idea with RDSA is to use the basic SA recursion in (6.5), but to replace the FD gradient approximation in (6.6) with a more efficient gradient approximation generated with the help of a Monte Carlo-generated perturbation vector. The basic form of this approach requires only two loss measurements to approximate the gradient vector (for any dimension p) and replaces the deterministic perturbations of FDSA (i.e., the $\pm c_k\boldsymbol{\xi}_i$ at each k for all $i = 1, 2,\ldots, p$) with *random* perturbations. Relative to classical two-sided FDSA ((6.5) and (6.6)), RDSA provides the potential for increased efficiency because of the p-fold reduction in loss measurements per iteration ($2p$ for FDSA versus 2 for

RDSA). (Koronacki, 1975, also considers the idea of random perturbations, including some convergence theory. This paper suggests the use of $2m$ loss measurements, with m usually less than p. There is, however, no formal evidence of improved efficiency over basic FDSA.)

The basic form for the ith component of the RDSA gradient approximation at iteration k is

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

where $\pi_k = [\pi_{k1}, \pi_{k2}, \ldots, \pi_{kp}]^T$ is a vector of Monte Carlo-generated random variables satisfying certain regularity conditions. Although we wait until Chapter 7 to analyze the properties and implications of this type of approach in detail, note that the two $y(\cdot)$ values are reused for all elements of the gradient approximation. Ermoliev (1969) includes analysis of the bias in the gradient approximation and considers one specific distribution (uniform) for the π_{ki}.

Polyak and Tsypkin (1973) and Kushner and Clark (1978, Sect. 2.3.5) perform a more complete theoretical analysis of RDSA-type algorithms, including a demonstration of a.s. convergence. These works consider the RDSA form with perturbations that are uniformly distributed on a p-dimensional sphere. Convergence alone, however, is not a compelling reason to use the RDSA methods, since the FDSA algorithm also converges using the same type of input information (noisy loss measurements). Nevertheless, the p-fold savings in loss measurements *per iteration* provides a tantalizing suggestion that RDSA may provide overall savings *across iterations* in the optimization process if p is large.

Using asymptotic theory, Kushner and Clark (1978, Sects. 2.3.5 and 7.4) provide the first rate-of-convergence analysis for RDSA with the aim of rigorously addressing the efficiency question. They draw a mainly negative conclusion about the efficiency gains of RDSA over FDSA: "... it is conceivable that with such a method [RDSA] the rate of convergence, as a function of the total number of observations used, might be increased [relative to FDSA] for large [dimension]. This hope is somewhat in vain..." (Kushner and Clark, 1978, p. 58). This conclusion, however, may be misleading, appearing to result from an analysis too restrictive in the choice of the gains (using only asymptotically optimal decay rates for the gains) and (generally) too conservative in the bound applied to the bias in the gradient estimate. In fact, significant improvements are possible in practice with this or similar (simultaneous perturbation) algorithms.

Chapter 7 is devoted to an extensive study of SA algorithms with Monte Carlo randomness. The simultaneous perturbation SA method considered there has a gradient approximation with an algebraic form similar to the RDSA form above. More importantly, the chapter considers general perturbation distributions and the associated regularity conditions. The choice of probability distribution for the Monte Carlo perturbations is important in realizing the desired improvements in efficiency.

6.9 CONCLUDING REMARKS

The stochastic approximation framework introduced in Chapters 4 and 5 is a powerful approach for dealing with nonlinear root-finding and optimization problems, especially problems involving noisy measurements of the functions involved. This chapter has continued with the SA-based *optimization* setting emphasized in Chapter 5, but with the significant difference that direct measurements of the gradient are not required. Rather, this chapter has considered methods that build gradient approximations from measurements of the loss function. This *mathematical* distinction has profound *practical* implications.

In particular, it is often difficult or impossible to calculate the stochastic gradient for the methods of Chapter 5. We discussed three examples—parameter estimation with complicated loss functions, model-free control, and simulation-based optimization—illustrating the difficulties. It is in such cases that the FD-based method of this chapter is most useful. While the direct search methods of Chapter 2 are also based on only loss function values (no gradients), their implementation with *noisy* loss measurements is largely ad hoc. This lack of formal justification with noisy measurements also applies to standard implementations of other popular methods, including simulated annealing and genetic algorithms (Chapters 8–10). In contrast, the FDSA method has a deep theoretical justification with noisy measurements.

One of the main shortcomings of FDSA is its inefficiency in high-dimensional problems. That is, the number of loss measurements in each gradient approximation grows directly with the dimension. Because a typical implementation for optimization requires many gradient approximations (one at each iteration), the overall number of loss measurements in the optimization process may become prohibitive. To tackle this problem, some methods have been introduced that create gradient approximations via Monte Carlo schemes. By reducing the number of loss measurements for each gradient approximation, these methods may offer significant improvements in efficiency, as discussed in detail in Chapter 7.

EXERCISES

6.1 Derive the multivariate analogue of (6.4) (i.e., the chain rule form for $\partial Q_k/\partial \boldsymbol{\theta}$) with the relevant gradient expressions expressed in matrix-vector notation (as in Appendix A) when $\boldsymbol{x}_k \in \mathbb{R}^q$ and $\boldsymbol{u}_k \in \mathbb{R}^r$ for some $q > 1$, $r > 1$. Use Q_k as given in (6.2).

6.2 Show that the one-sided FD gradient estimator has an $O(c_k)$ bias under the same conditions used in deriving the $O(c_k^2)$ bias of the two-sided version (eqn. (6.8)). Give at least two reasons why this does not necessarily mean that the one-sided estimator is inferior in practice.

6.3 Consider the function $L(\boldsymbol{\theta}) = t_1^4 + t_1^2 + t_1 t_2 + t_2^2$, $\boldsymbol{\theta} = [t_1, t_2]^T$. Suppose that $\hat{\boldsymbol{\theta}}_k = [1, 1]^T$ for some k. Based on the FD approximation in (6.6), determine

upper bounds to the magnitudes of the components of the bias $\boldsymbol{b}_k$ when $c_k = 0.2$ and $c_k = 0.5$. Comment on these magnitudes relative to the magnitudes of the corresponding components of $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$; are the biases potentially significant?

6.4 It was noted in Section 6.5 that $\lim_{k\to\infty} E[k^{\beta/2}(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*)]$ is not necessarily equal to $\boldsymbol{\mu}_{FD}$ when $k^{\beta/2}(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*) \xrightarrow{\text{dist.}} N(\boldsymbol{\mu}_{FD}, \boldsymbol{\Sigma}_{FD})$. Explain this in light of the results in Appendix C.

6.5 Show that gain conditions B.1′, $\beta > 0$, and $3\gamma - \alpha/2 \geq 0$ imply the relationships in (6.14).

6.6 Condition A.1′ (Section 6.4) includes the gain condition $\sum_{k=0}^{\infty} a_k c_k < \infty$. How does A.1′ further restrict (6.14)?

6.7 Let $\boldsymbol{W}_k = k^{-\gamma}\{\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) - E[\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)\,|\,\Im_k]\}$. Assume that the noises ε in the measurements $y(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \varepsilon$ are i.i.d. with $\text{var}(\varepsilon) = \sigma^2/2$. A key step in the proof of the asymptotic normality in (6.13) is to show that $E(\boldsymbol{W}_k \boldsymbol{W}_k^T | \Im_k) \to c^{-2}\sigma^2 \boldsymbol{I}_p/4$ (a.s.) as $k \to \infty$, where c appears in the numerator of c_k in (6.11). Sketch the arguments showing that this is true; state any additional conditions needed. (Note: This result also holds under more general conditions on the noise ε.)

6.8 Repeat the study described in Example 6.5 and statistically compare the four sample mean results you obtain with the four sample means in Table 6.1. (Hint: Use the appropriate two-sample t-test from Appendix B after "backing out" the approximate sample standard deviations from the confidence intervals shown in Table 6.1.)

6.9 Provide some intuitive rationale for the practical guideline that the numerator c in $c_k = c/(k+1)^\gamma$ should be set at approximately the magnitude of the initial standard deviation of the measurement noise in $y(\boldsymbol{\theta})$.

6.10 Given a desirable step size of 0.1 for the elements of $\boldsymbol{\theta}$, use the semiautomatic method of Section 6.6. Show that the values of c, A, and a in Examples 6.6 and 6.7 are $c = 1$, $A = 5$ (using 10 percent of iterations), and, approximately, $0.20 \leq a \leq 0.25$ ($a = 0.25$ is used to generate the relevant curve in Figure 6.2 of Example 6.6 and the relevant column in Table 6.2 in Example 6.7).

6.11 The following coefficient values are used in Examples 6.6 and 6.7:

- FDSA with semiautomatic gains: see Exercise 6.10.
- FDSA with manual tuning: same as Exercise 6.10 except that $a = 0.5$.
- Random search method B: 20 loss measurements are averaged at each iteration, and normally distributed perturbations are used with distribution $N(\boldsymbol{0}, 0.5^2 \boldsymbol{I}_{10})$.

Produce a plot analogous to Figure 6.2 (i.e., three curves, each an average of 50 independent runs), but rather than show loss values, show the sample mean of the normalized errors $\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\| / \|\hat{\boldsymbol{\theta}}_0 - \boldsymbol{\theta}^*\|$ as a function of the number of loss measurements. Comment on the results relative to those in Figure 6.2.