# Chapter 6
# An Overview of Stochastic Approximation

Marie Chau and Michael C. Fu

**Abstract** This chapter provides an overview of stochastic approximation (SA) methods in the context of simulation optimization. SA is an iterative search algorithm that can be viewed as the stochastic counterpart to steepest descent in deterministic optimization. We begin with the classical methods of Robbins–Monro (RM) and Kiefer–Wolfowitz (KW). We discuss the challenges in implementing SA algorithms and present some of the most well-known variants such as Kesten's rule, iterate averaging, varying bounds, and simultaneous perturbation stochastic approximation (SPSA), as well as recently proposed versions including scaled-and-shifted Kiefer–Wolfowitz (SSKW), robust stochastic approximation (RSA), accelerated stochastic approximation (AC-SA) for convex and strongly convex functions, and Secant-Tangents AveRaged stochastic approximation (STAR-SA). We investigate the empirical performance of several of the recent algorithms by comparing them on a set of numerical examples.

## 6.1 Introduction

Stochastic approximation (SA) is a recursive algorithm that can be viewed as the stochastic counterpart to steepest descent in deterministic optimization. SA was introduced by Robbins and Monro in 1951 [32] to solve noisy root-finding problems and was later applied to the setting of stochastic optimization by solving for the zero of the gradient. The gradient-free setting was addressed by Kiefer and Wolfowitz in 1952 [24]. SA is currently one of the most widely applicable and most useful methods for simulation optimization.

Consider the stochastic optimization problem

$$\min_{x \in \Theta} f(x), \tag{6.1}$$

M. Chau • M.C. Fu (✉)
University of Maryland, College Park, MD, USA
e-mail: mchau@math.umd.edu; mfu@umd.edu



M.C. Fu (ed.), *Handbook of Simulation Optimization*, International Series in Operations Research & Management Science 216, DOI 10.1007/978-1-4939-1384-8_6

where $f(x) = \mathsf{E}[Y(x,\xi)]$ is a performance measure, $Y(x,\xi)$ is a sample performance, $\xi$ denotes the stochastic effects, and $\Theta \subseteq \mathbb{R}^d$ is a continuous parameter space. In this case, the objective is to find a sequence $\{x_n\}$ that converges to a unique (local) optimum

$$x^* = \arg\min_{x \in \Theta} f(x), \tag{6.2}$$

by using the recursion

$$x_{n+1} = \Pi_\Theta \left( x_n - a_n \hat{\nabla} f(x_n) \right), \tag{6.3}$$

where $\Pi_\Theta(x)$ is a projection of $x$ back into the feasible region $\Theta$ if $x \notin \Theta$, $a_n > 0$ is the step size or gain size, $\hat{\nabla} f(x_n)$ is an estimate of the gradient $\nabla f(x_n)$, and $x_N$ is the output, where $N$ is the stopping time, which we denote by $x_N^*$. The projection operator is only required in the constrained setting. Moreover, the minimization problem in (6.1) and (6.2) could easily be changed to maximization by changing the sign of $a_n$ in (6.3). The two classical methods, Robbins–Monro (RM) and Kiefer–Wolfowitz (KW), estimate $\nabla f(x_n)$ using unbiased direct gradient estimates and finite difference gradient estimates, respectively. Under certain conditions, RM and KW have the respective asymptotic convergence rates $O(n^{-1/2})$ and $O(n^{-1/3})$.

Advances in SA have included the development of new algorithms, modifications to existing ones, and new asymptotic and finite-time theory. Asymptotic convergence properties of KW, RM and their variations have been a major research focus (cf. [12, 14, 15, 28, 31, 41]). The original RM and KW algorithms apply to one-dimensional problems, but they were later extended to the multidimensional case [2]. Furthermore, the earlier conditions used to prove convergence for RM and KW were relaxed to obtain almost sure convergence [2]. The estimate $x_n$ in (6.3) was shown to be asymptotically normal [15] with the optimal convergence rate of $O(n^{-1/2})$ [8]. More recently, researchers have placed greater emphasis on finite-time theory as well as error bounds on the difference between objective value at the current estimate and the optimal objective value (i.e., $\mathsf{E}[f(x_N^*) - f(x^*)]$), which the next chapter treats in more detail.

Although recursion (6.3) is quite simple, the choice of step-size sequence $\{a_n\}$, gradient estimator $\hat{\nabla} f(x_n)$, projection operator $\Pi_\Theta$, and output $x_N^*$ each has a significant impact on the performance of the algorithm.

We first discuss the influence of step-size sequence $\{a_n\}$ on the finite-time performance. It is widely known that the practical performance of the classical RM and KW algorithms is highly dependent on the choice of $\{a_n\}$ and often performs poorly without tuning. The algorithm can experience a long oscillatory period if the gain sequence $\{a_n\}$ is "too large," where the iterates jump back and forth without approaching the optimum $x^*$, which can be seen in Fig. 6.1 (left graph), or a degraded convergence rate if $\{a_n\}$ is "too small" relative to the magnitude of the gradient, where the iterates barely move, which can be seen in Fig. 6.1 (right graph) (only $x_1$ is labeled since the other iterates are in the same position).

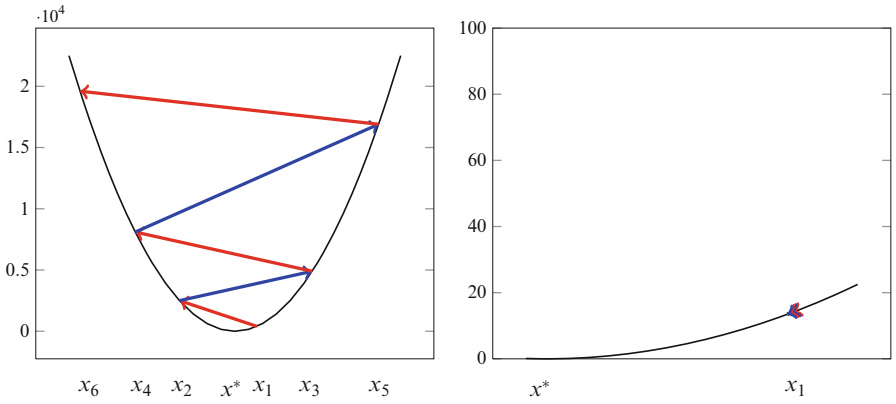


**Fig. 6.1** Sensitivity of SA to step size $a_n$ when $a_n$ is "too large" relative to the gradient (left graph) and when $a_n$ is "too small" relative to the gradient (right graph)

One approach to tackle the sensitivity is to develop robust step-size sequences, i.e., an adaptive step-size rule. The earliest attempt at adaptive step sizes was Kesten's rule, which can be applied to both RM and KW [23]. This step size only decreases when there is a directional change in the iterates, i.e., $(x_{n+1} - x_n)(x_n - x_{n-1}) < 0$. The idea behind this adaptive step size is that, if the iterates move in the same direction, there is reason to believe they are not in close proximity of the optimum, so the momentum should not be decreased. Later, this idea was extended by increasing the step size, as opposed to keeping it constant when the consecutive errors in the estimates are of the same sign, to increase the speed to convergence [37]. A recent attempt, called scaled-and-shifted Kiefer–Wolfowitz (SSKW) and described in more detail in Sect. 6.4.1, adaptively adjusts the step-size sequence $\{a_n\}$ finitely many times during the course of a modified version of KW [4]. The rationale behind the procedure is to increase the gain size so the iterates are able to move from one endpoint to the other in the one-dimensional case (ideas are analogous in the multidimensional case [3]), which ensures the step sizes are large enough to make noticeable progress towards the optimum in finite-time. If the step sizes $\{a_n\}$ are too large, then they decrease at a faster pace during the recourse stage. Another method used to select an adaptive step size is generating an approximation of the inverse of the Hessian, which is the stochastic analogue to the deterministic Newton–Raphson method [42]. According to Yakowitz et al. [45], "…the optimal choice [of step-size sequence] involves the Hessian of the risk [objective] function, which is typically unknown and hard to estimate." Hessians have been estimated using a set of finite difference gradient approximations [16], heuristics based on quasi-Newton methods [22], and finite difference approximations using gradient estimates [33, 42]. The choice of $\{a_n\}$ has a significant impact on the finite-time performance of the algorithm, which is quite difficult to characterize theoretically.

Another approach to reduce the sensitivity of the estimated optimum to $\{a_n\}$ is to modify the output so that the algorithm puts less emphasis on the last iterate. The underlying idea behind methods of this type is to take longer steps and incorporate a subset of the iterates into the output to decrease the reliance on the last iterate. Iterate averaging, which takes the average of all iterates as the output as opposed to the final iterate, was the earliest proposal [31, 34]. This algorithm can be easily implemented, is "robust," since it is less sensitive to the initial step size choice, and exhibits $O(n^{-1/2})$ asymptotic convergence rate under appropriate conditions. A generalization of iterate averaging, called robust stochastic approximation (RSA) algorithm [30], uses the step-size sequence $\{a_n\}$ for weighting the iterates, which will be described in more detail in Sect. 6.4. Another generalization introduces a proximity function into the objective function, which acts as a regularization term to prevent the next iterate update $x_{n+1}$ from being too far from $x_n$. An example from this class of algorithms called the accelerated stochastic approximation (AC-SA) algorithm [20] is detailed in Sect. 6.4.

Asymptotic theory for SA initially only considered functions satisfying specific global conditions; however, it is only necessary for the requirements to hold for a compact set as long as it contains the optimum, so the projection operator is particularly important in the constrained optimization setting. The feasible region $\Theta$ must be large enough to increase the likelihood that $x^* \in \Theta$, but enlarging the search space may deteriorate the performance of the algorithm. Adaptively increasing the search space still leads to provable convergence with an appropriate projection operator, such as adaptively projecting the iterates onto an increasing compact set [1].

The gradient estimate is also clearly central to any SA algorithm, and thus the subject of stochastic gradients is treated in depth in Chap. 5. The most common gradient estimate is obtained using finite differences, because it only requires performance measures and no other additional information from the system. For each dimension, the finite difference gradient estimate requires two performance measures, and if the measurements are highly volatile, then the gradient estimates can be noisy. Furthermore, finite difference estimates become computationally expensive in high dimensions, since the cost grows linearly with the parameter dimension [2]. Simultaneous perturbation stochastic approximation (SPSA) [38] only requires two estimates of the objective function to approximate the gradient, and the computational cost is independent of the dimension of the parameter space. Recently, a new SA algorithm called Secant-Tangents AveRaged stochastic approximation (STAR-SA) has been proposed, and it employs a hybrid gradient estimator that combines direct gradient estimates with a symmetric finite difference gradient estimate [5, 6].

The remainder of the chapter is organized as follows. We introduce the classical stochastic approximation methods, Robbins–Monro (RM) and Kiefer–Wolfowitz (KW) in Sect. 6.2. In Sect. 6.3, we present some of the most useful enhancements for simulation optimization: Kesten's rule, Ruppert–Polyak averaging iterates, varying bounds, and SPSA. We describe four recent developments, scaled-and-shifted

Kiefer–Wolfowitz (SSKW), robust stochastic approximation (RSA), accelerated stochastic approximation (AC-SA) for convex and strongly convex functions, and Secant-Tangents AveRaged (STAR) stochastic approximation in Sect. 6.4. In Sect. 6.5, we present numerical experiments comparing KW-type algorithms (original KW, KW with Kesten's rule, SSKW), RM-type methods (original RM, RM with iterate averaging, RSA, AC-SA), and single versus mixed gradients (RM, KW, STAR-SA). Finally, we close with concluding remarks in Sect. 6.6.

## 6.2 Classical Methods

The classical RM and KW algorithms address unconstrained stochastic optimizations problems, so we consider the recursive scheme

$$x_{n+1} = x_n - a_n \hat{\nabla} f(x_n), \tag{6.4}$$

which is identical to (6.3) with the exception of the projection operator.

### *6.2.1 Robbins–Monro (RM) Algorithm*

The RM algorithm was introduced to solve the root-finding problem

$$M(x) = \alpha$$

for $x \in \mathbb{R}$, where $M(x)$ is a monotone function and $\alpha \in \mathbb{R}$. However, it was later applied to a specific case of root-finding in the stochastic optimization setting, where the objective is to optimize a stochastic objective function $f(x)$ by setting $M(x) = \nabla f(x)$ and $\alpha = 0$. RM solves this problem iteratively as in (6.3) by replacing $\hat{\nabla} f(x_n)$ with an unbiased estimator, and the output is taken as the last iterate, $x_N^*$, where $N$ is the stopping time. However in RM, the direct gradient measurements are still approximations to the actual gradient because of the presence of noise ($\hat{\nabla} f(x_n) = \nabla f(x_n) + \varepsilon_n$, where $\varepsilon_n$ is noise with zero mean). Given the appropriate parameters, this algorithm converges asymptotically at a rate of $O(n^{-1/2})$ [35].

**Theorem 6.1 (Theorem 2 [32]).** *Assume $\nabla f(x)$ has a unique root $x^*$ and suppose $\hat{\nabla} f(x)$ is an unbiased gradient estimator, i.e., $\nabla f(x) = \mathsf{E}[\hat{\nabla} f(x)]$. If the sequence $\{x_n\}$ is generated from (6.4) and the following conditions hold:*

1. *$\{a_n\}$ is a sequence of positive constants such that $\sum_{n=1}^{\infty} a_n = \infty$ and $\sum_{n=1}^{\infty} a_n^2 < \infty$.*
2. *$\nabla f(x) \geq 0$ for $x > x^*$ and $\nabla f(x) \leq 0$ for $x < x^*$.*
3. *There exists a positive constant $C$ such that $\mathsf{P}(|\hat{\nabla} f(x)| \leq C) = 1 \ \forall x$.*

*Then $x_n \xrightarrow{L^2} x^*$ as $n \to \infty$, where $\xrightarrow{L^2}$ denotes mean-squared convergence.*

The most well-known conditions are restrictions on the gain sequence $\{a_n\}$. Generally, $a_n \to 0$ but $\sum_{n=1}^{\infty} a_n = \infty$, which prevents the step size from converging

to zero too quickly, so the iterates are able to make progress to $x^*$ and do not get stuck at a poor estimate. The usual form is $a_n = \frac{\theta_a}{(n+A)^\alpha}$, where $\theta_a > 0$, $A \geq 0$ and $\frac{1}{2} < \alpha \leq 1$, with $A = 0$ and $\alpha = 1$ as a commonly used choice. The objective function $f$ is assumed to have a global minimum with a bounded derivative.

### 6.2.2 Kiefer–Wolfowitz (KW) Algorithm

The KW stochastic approximation algorithm is referred to as a gradient-free or stochastic zeroth-order method in the following chapter, since it only requires noisy measurements of the function and does not require additional information on the system dynamics or input distributions. The original KW iterative scheme

$$x_{n+1} = x_n - a_n \frac{Y(x_n + c_n, \xi_n^+) - Y(x_n - c_n, \xi_n^-)}{2c_n}, \tag{6.5}$$

estimates the gradient using a symmetric finite difference gradient estimate, and under certain conditions, KW can achieve an asymptotic convergence rate of $O(n^{-1/3})$. In addition, common random numbers (CRN) can be employed to decrease the variance of estimates, and KW can achieve an asymptotic convergence rate of $O(n^{-1/2})$ in certain settings [25].

**Theorem 6.2 (Theorem in [24]).** *Assume $f(x) = \mathsf{E}[Y(x,\xi)]$. If the sequence $\{x_n\}$ is generated from (6.5) and the following conditions hold:*

1. *Let $\{a_n\}$ and $\{c_n\}$ be positive tuning sequences satisfying the conditions*

$$c_n \to 0, \ \sum_{n=1}^{\infty} a_n = \infty, \ \sum_{n=1}^{\infty} a_n c_n < \infty, \ \sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty.$$

2. *$f(x)$ is strictly decreasing for $x < x^*$, strictly increasing for $x > x^*$.*
3. *$\mathrm{Var}[Y(x,\xi)] < \infty$ and the following regularity conditions hold:*

   *1) There exist positive constants $\beta$ and $B$ such that*

$$|x' - x^*| + |x'' - x^*| < \beta \Longrightarrow |f(x') - f(x'')| < B|x' - x''|.$$

   *2) There exist positive $\rho$ and $R$ such that*

$$|x' - x''| < \rho \Longrightarrow |f(x') - f(x'')| < R.$$

   *3) For every $\delta > 0$ there exists a positive $\pi(\delta)$ such that*

$$|x - x^*| > \delta \Longrightarrow \inf_{\frac{\delta}{2} > \varepsilon > 0} \frac{|f(x+\varepsilon) - f(x-\varepsilon)|}{\varepsilon} > \pi(\delta).$$

*Then $x_n \xrightarrow{p} x^*$ as $n \to \infty$, where $\xrightarrow{p}$ denotes convergence in probability.*

Condition 1 assures that the step size $a_n$ does not converge to zero too fast, so the iterates do not get stuck at a poor estimate. In addition, the condition restricts the finite difference step size $c_n$ from decreasing too quickly, which constrains the noise of the gradients. The second condition insures that there is a global optimum. The first regularity condition requires $f(x)$ to be locally Lipschitz in a neighborhood of $x^*$; the second one prevents $f(x)$ from changing drastically in the feasible region; and the last one prohibits the function from being very flat outside a neighborhood of $x^*$ so that the iterates approach the optimum. Although the KW algorithm converges asymptotically, its finite-time performance is dependent on the choice of tuning sequences, $\{a_n\}$ and $\{c_n\}$. If the current $x_n$ is in a relatively flat region of the function and the $a_n$ is small, then the convergence will be slow. On the other hand, if the $x_n$ is located in a very steep region of the function and $\{a_n\}$ is large, then the iterates will experience a long oscillation period. If $\{c_n\}$ is too small, the gradient estimates using finite differences could be extremely noisy.

KW has been extended to higher dimensions, and two common gradients considered are symmetric differences and forward differences whose $i$th component is given by

$$\hat{\nabla} f_i(x_n) = \begin{cases} \dfrac{Y(x_n + c_n e_i, \xi_{n,i}^+) - Y(x_n - c_n e_i, \xi_{n,i}^-)}{2c_n} & \text{symmetric difference,} \\ \dfrac{Y(x_n + c_n e_i, \xi_{n,i}^+) - Y(x_n, \xi_{n,i})}{c_n} & \text{one-sided forward difference,} \end{cases}$$

where $e_i$ denotes a $d$-dimensional $i$th unit basis vector, $c_n \in \mathbb{R}^+$, and $Y(x, \xi)$ is an unbiased estimate of $f(x)$. This method perturbs each component of $x_n$ (i.e., $x_{n,i}$ for $i = 1, \ldots, d$) one at a time while holding all others constant and returns a corresponding function value estimate. For instance, symmetric differences requires the estimate of two function values $f(x_n + c_n e_i)$ and $f(x_n - c_n e_i)$ for $i = 1, \ldots, d$, and forward differences requires $f(x_n)$ and $f(x_n + c_n e_i)$ for $i = 1, \ldots, d$; therefore, using symmetric and one-sided forward difference estimates involves $2d$ and $d + 1$ simulation replications, respectively. Although using the symmetric difference scheme is computationally more expensive, it has the potential to reach an asymptotic convergence rate of $O(n^{-1/3})$ compared to $O(n^{-1/4})$ for forward differences. For $d = 1$, the computational cost is identical for both the symmetric difference and one-sided forward difference. Compared with the RM algorithm, however, KW convergence rates are typically inferior, although under certain conditions with CRN $\xi_{n,i}^+ = \xi_{n,i}^-$, KW algorithms also can achieve the $O(n^{-1/2})$ asymptotic convergence rate. For simulation optimization, RM is not always applicable since additional information is needed, which may not be readily available or is difficult to obtain. For KW, there is an additional task of appropriately choosing the difference sequence $\{c_n\}$. In general, KW is a simple algorithm to implement for simulation optimization applications, albeit costly in high-dimensional settings.

## 6.3 Well-Known Variants

In this section, we elaborate on Kesten's rule, iterate averaging, adaptively varying bounds, and SPSA.

### *6.3.1 Kesten's Rule*

It is well-known that the classical SA algorithms are extremely sensitive to the step-size sequence $\{a_n\}$. Therefore, it could be advantageous to consider adaptive step sizes that adjust based on the ongoing performance of the algorithm, in hopes of adapting them to the characteristics of the function at the current location of the iterate and proximity of the current iterate to the optimum. Kesten's rule [23] decreases the step size only when there is a directional change in the iterates. The notion behind this adaptive step size is that, if the iterates continue in the same direction, there is reason to believe they are approaching the optimum and the pace should not be decreased in order to accelerate the convergence. If the errors in the estimate values change signs, it is an indication that either the step size is too large and the iterates are experiencing long oscillation periods or the iterates are in the vicinity of the true optimum; either way, the step size should be reduced to a more appropriate step size or to hone in on $x^*$. The following algorithm is for the one-dimensional case $d = 1$.

---

**SA Algorithm Using Kesten's Rule**

- Input. Choose $x_1 \in \Theta, \{a_n\}$, $\Pi_\Theta$, and stopping time $N$.
- Initialize.
  - Let $n = 2$ and $k = 1$.
  - Generate an estimate $\hat{\nabla} f(x_1)$ of $\nabla f(x_1)$.
  - Compute $x_2 = \Pi_\Theta(x_1 - a_1 \hat{\nabla} f(x_1))$.
- While $n < N$,
  - Step 1. Generate an estimate $\hat{\nabla} f(x_n)$ of $\nabla f(x_n)$.
  - Step 2. Compute $x_{n+1} = \Pi_\Theta(x_n - a_k \hat{\nabla} f(x_n))$. If $(x_{n+1} - x_n)(x_n - x_{n-1}) < 0$, go to Step 3. Otherwise, go to Step 4.
  - Step 3. Let $n = n + 1$ and $k = k + 1$. Go to Step 1.
  - Step 4. Let $n = n + 1$. Go to Step 1.
- Output. $x_N^* = x_N$.

---

Kesten's rule can be applied to both RM and KW and still guarantee convergence in probability, as long as $\{a_n\}$ satisfies condition (1) in Theorems 1 and 2 for RM and KW, respectively [23]. An extension of Kesten's rule to higher dimensions is discussed in [11]. See [18] for an extensive review of both deterministic and stochastic step sizes.

### 6.3.2 Averaging Iterates

Iterate averaging approaches SA from a different angle. Instead of fine-tuning the step sizes to adapt to the function characteristics, iterate averaging takes bigger steps (i.e., $a_n$ larger than $O(n^{-1})$) for the estimates to oscillate around the optimum, so the average of the iterates will result in a good approximation to the true optimum. The idea is simple, and yet can be very effective. It is easy to see that for this method to be successful, it is essential for the iterates to surround the optimum in a balanced manner, and the domain for which the iterates oscillate shrinks as $n$ increases. Averaging trajectories reduces the sensitivity to the initial step size choice. The algorithm follows recursion (6.3) for the RM case; however, instead of the taking the last iterate $x_N$ as the output, the optimum is estimated by

$$x_N^* = \frac{1}{N}\sum_{n=1}^{N} x_n,$$

which is an average of $N$ iterates, where $N$ is the stopping time. Under "classic" assumptions, iterate averaging achieves the same convergence rate as the RM method. Furthermore, $\sqrt{n}(x_n^* - x^*)$ is asymptotically normal with mean zero and the smallest covariance matrix, which is the inverse of the average Fisher information matrix. (cf. [31]). A constant step size can be applied and yields convergence in distribution [28].

A variation of this method is called the "sliding window" average, which is based on the last $m$ iterates:

$$x_N^* = \frac{1}{m}\sum_{n=N-m+1}^{N} x_n. \tag{6.6}$$

An advantage of (6.6) is it ignores the first $N-m$ iterates, which may be poor estimates, since the first iterate is arbitrary, and averages only the last $m$, which are assumed to be closer to $x^*$. Asymptotic normality for a growing window is shown in [26, 28], which also includes constant step sizes. Another modification of the original method incorporates $x_N^*$ with $x_N$ in the components being averaged, which is known as the feedback approach [27]. These methods are suited for problems where the iterates hover around the optimum. In an empirical study, iterate averaging was applied to SPSA [29]. The results suggest that if the Hessian of $f(x)$ is large,

averaging is considered ideal, since it is associated with a high variability in $f(x)$, which indicates the iterates are moving around the optimum. In general, averaging iterates leads to more robustness with respect to step-size sequence because of the reduced sensitivity, while converging at the same optimal asymptotic rate as RM. Inspired by iterate averaging, weighted averages for KW was presented to achieve the optimal asymptotic convergence rate $O(n^{-1/2})$ under certain conditions [13]. Under certain parameter settings, iterate averaging and weighted averaging produce the same estimator.

### *6.3.3 Varying Bounds*

Initially, the asymptotic theory for SA only considered functions satisfying specific global conditions; however, subsequently it was shown the requirements need only hold on a compact set $\Theta \in \mathbb{R}^d$ containing the optimum. Therefore, the projection operator is particularly important in the constrained optimization setting. Since the optimum is unknown, the compact set should be large enough so that $x^* \in \Theta$ with high probability; however, this may increase the potential of an algorithm to perform poorly due to the size of the parameter search space [1] . For instance, if the compact set is very large, the step size is extremely small, and the current iterate is extremely far from the optimum, then the convergence is likely to be slow; however, if the compact set is small and contains the optimum, then the iterates will never be too far from the optimum. Even if the step sizes are small, the convergence will be much faster in comparison to the algorithm restricted to a much larger set.

One of the first ideas was to project the iterates onto a predetermined fixed point once the magnitude of the iterate surpassed an arbitrarily specified threshold, with the threshold increasing after it is exceeded [10]. This method converges asymptotically, but in practice, it has its pitfalls. When an iterate is projected onto an arbitrary fixed point, in a sense, the algorithm restarts from this "initial" value with a smaller sequence of step sizes. Not only does it lose all of the progress gained from the iterations prior to the projection, but the reduction in step size could hinder the convergence by moving even slower towards the optimum. To circumvent this issue, it was shown that it suffices to project the iterates onto a predetermined bounded set [46]. This is a slight improvement, since the iterates do not start from the same position with an even smaller step size. However, it still has its limitations, since the initial start values are restricted to the predetermined compact set. Later, an algorithm defined over a growing feasible region by writing $\Theta$ as an increasing sequence of compact sets (i.e., $\Theta_m \subseteq \Theta_{m+1}$, where $\Theta = \cup_m \Theta_m$) was introduced [1]. The orthogonal projection operator changes from $\Pi_{\Theta_m}$ to $\Pi_{\Theta_{m+1}}$ if $x_n \notin \Theta_m$. The idea is to start with a smaller feasible region $\Theta_1$ and only increase when there is reason to believe the optimum $x^* \notin \Theta_1$ (i.e., when the $x_n \notin \Theta_1$). Since the projection is made onto the current compact set $\Theta_m$, the progress gained up to that point is not lost. The feasible region $\Theta$ is written as $\cup_{m=1}^{\infty} \Theta_m$, so it is impossible for $x^* \notin \Theta_m$ for

some $m$. If $x^*$ is contained in one of the earlier compact sets and if they grow slowly, the empirical results could improve significantly. The key in the performance is to choose the sequence $\{\Theta_m\}$ appropriately. If it grows too quickly, the results might be very similar to that of the original SA algorithm. The following algorithm and convergence result are for the RM multidimensional case $d \geq 1$, where $||\cdot||$ denotes the Euclidean norm.

---

**SA with Varying Bounds**

- Input. Choose $x_1 \in \Theta_1$, $\{a_n\}$ and $\{\Theta_m\}$.
- Initialize. Let $n = 1$ and $m = 1$.
- While $n < N$,
  - Step 1. Generate an estimate $\hat{\nabla} f(x_n)$ of $\nabla f(x_n)$.
  - Step 2. Compute $x'_{n+1} = x_n - a_n \hat{\nabla} f(x_n)$. If $x'_{n+1} \in \Theta_m$, go to Step 3. Otherwise, go to Step 4.
  - Step 3. Let $x_{n+1} = x'_{n+1}$, $n = n+1$ and go to Step 1.
  - Step 4. Let $x_{n+1} = \Pi_{\Theta_m}(x'_{n+1})$, $n = n+1$, $m = m+1$ and go to Step 1.
- Output. $x_N^* = x_N$.

---

**Theorem 6.3 (Theorem 2 [1]).** *Let the sequence $\{x_n\}$ be generated using the above algorithm, $\varepsilon_n = \hat{\nabla} f(x) - \mathsf{E}[\hat{\nabla} f(x)|\mathscr{F}_n]$, and $\beta_n = \mathsf{E}[\hat{\nabla} f(x)|\mathscr{F}_n] - \nabla f(x)$, where $\mathscr{F}_n$ is the smallest $\sigma$-algebra used to generate $x_{n+1}$. If the following conditions hold:*

1. *The sequence $\{\Theta_m\}$ is a set of compact convex sets such that $\Theta_m \subseteq \Theta_{m+1}$ for all $m$ and $\cup_{m=1}^{\infty} \Theta_m = \Theta$.*
2. *The positive sequences of real numbers $\{a_n\}$ and $\{c_n\}$ converge to zero such that $\sum_{n=1}^{\infty} a_n = \infty$, $\sum_{n=1}^{\infty} a_n c_n < \infty$, and $\sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty$.*
3. *There exists $\kappa \geq 0$ such that $\mathsf{E}[||\varepsilon_n||^2|\mathscr{F}_n] \leq \frac{\kappa}{c_n^2}(1 + ||x_n - x^*||^2)$ a.s. for all $n$.*
4. *$||\beta_n||$ is bounded a.s. for all $n$, and $\sum_{n=1}^{\infty} a_n ||\beta_n|| < \infty$ a.s.*
5. *There exist a positive sequence of real numbers $\{M_n\}$ and integer $N \geq 1$ such that $\sum_{n=1}^{\infty} a_n^2 M_n^2 < \infty$ and for all $n \geq N$, $\sup_{x \in \Theta_{n-1}} ||f(x)|| \leq M_n$.*
6. *There exists a unique $x^* \in \Theta$ such that $\nabla f(x^*) = 0$, and for all $0 < \delta \leq 1$, $\inf_{x \in \Theta : \delta \leq ||x - x^*|| \leq \delta^{-1}} f(x)^{\mathsf{T}}(x - x^*) > 0$.*

*Then $x_n \to x^*$ a.s. as $n \to \infty$.*

If an appropriate increasing sequence of compact sets is chosen, the finite-time performance can improve significantly, but this optimal choice is still an open problem.

### 6.3.4 Simultaneous Perturbation Stochastic Approximation (SPSA)

Simultaneous perturbation stochastic approximation (SPSA) specifically addresses multivariate optimization problems [38]. Similar to KW-type algorithms, SPSA only requires the objective function values to approximate the underlying gradient and is therefore easy to implement. However, SPSA only requires two functional evaluations at each iteration regardless of the dimension of the parameter space $\Theta$, which could potentially reduce the computational cost significantly in high-dimensional problems. SPSA perturbs the vector $x$ randomly in all directions simultaneously (hence, the name of the method) and the $i$th component of the gradient estimate has the form

$$\hat{\nabla} f_i(x_n) = \frac{Y(x_n + c_n\Delta_n, \xi_n^+) - Y(x_n - c_n\Delta_n, \xi_n^-)}{2c_n\Delta_{n,i}}, \tag{6.7}$$

where $\Delta_n = (\Delta_{n,1}, \ldots, \Delta_{n,d}) \in \mathbb{R}^d$ and generally assumed to be i.i.d. and independent across components, $c_n \in \mathbb{R}^+$ is the finite difference step size, and $\xi_n^{\pm}$ denotes the randomness. Observe that the numerator in (6.7) involves two function estimates and is identical for all $i$; therefore, the cost of the full gradient (aside from generating $\Delta_n$) is independent of dimension.

---

**SPSA Algorithm**

- Input. Choose $x_1 \in \Theta$, $\{a_n\}$, $\{c_n\}$, and stopping time $N$.
- Initialize. Let $n = 1$.
- While $n < N$,
  - Step 1. Generate a $d$-dimensional random perturbation vector $\Delta_n$.
  - Step 2. Generate an estimate of $\nabla f(x_n)$:

$$\hat{\nabla} f(x_n) = \frac{Y(x_n + c_n\Delta_n, \xi_n^+) - Y(x_n - c_n\Delta_n, \xi_n^-)}{2c_n} \begin{bmatrix} \Delta_{n,1}^{-1} \\ \vdots \\ \Delta_{n,d}^{-1} \end{bmatrix}$$

  - Step 3. Compute $x_{n+1} = x_n - a_n\hat{\nabla} f(x_n)$.
  - Step 4. Let $n = n + 1$. Go to Step 1.
- Output. $x_N^* = x_N$.

---

**Theorem 6.4 (Theorem 7.1 [40]).** *Suppose $f$ has a unique minimum $x^* \in \Theta$ and $\{x_n\}$ is generated using SPSA. If the following conditions hold:*

1. *The positive sequences of real numbers* $\{a_n\}$ *and* $\{c_n\}$ *converge to zero such that* $\sum_{n=1}^{\infty} a_n = \infty$ *and* $\sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty$.
2. *The function* $f(x) \in C^3$ *and bounded on* $\mathbb{R}^d$.
3. $||x_n|| < \infty$ *for all* $n$.
4. $\mathsf{E}[\varepsilon_n^+ - \varepsilon_n^- | \Delta_n, \mathscr{F}_n] = 0$ *and* $\mathsf{E}[(Y(x_n \pm c_n \Delta_n, \xi_n^{\pm})/\Delta_{n,i})^2]$ *is uniformly bounded for all* $n, i$.
5. $x^*$ *is an asymptotically stable solution of the differential equation* $\partial x(t)/\partial t = -\nabla f(x(t))$.
6. *For each* $n$, $\{\Delta_{n,i}\}_{i=1}^{d}$ *are identically distributed,* $\{\Delta_{n,i}\}$ *are independent and symmetrically distributed with zero mean and uniformly bounded in magnitude for all* $n, i$.

*Then* $x_n \to x^*$ *a.s. as* $n \to \infty$.

The optimal convergence rate for SPSA is $O(n^{-1/3})$ [38]. Various convergence proofs have been presented with slight modifications to the conditions (cf. [9,13,19, 38,43]). The perturbation sequence $\{\Delta_n\}$, where $\Delta_n = (\Delta_{n,1}, \ldots, \Delta_{n,d})$ with $\{\Delta_{n,i}\}$ independent, must have mean zero (i.e., $\mathsf{E}[\Delta_n] = 0$), and finite inverse moments (i.e., $\mathsf{E}[|\Delta_{n,i}|^{-1}] < \infty$ for $i = 1, \ldots, d$). As a result, the Gaussian distribution is not applicable. Instead, the most common distribution used is the symmetric Bernoulli taking a positive and negative value (i.e., $\pm 1$) with probability 0.5. In addition, an appropriately scaled $x_n$ is approximately normal for large $n$, and the relative efficiency of SPSA depends on the geometric shape of $f(x)$, choice of $\{a_n\}$ and $\{c_n\}$, distribution of $\{\Delta_{n,i}\}$, and noise level.

Many extensions to the original SPSA algorithm have been developed, e.g., the constrained setting using projection operators [17, 36]. A slight modification is the averaging of the SPSA gradient estimators. Instead of generating one gradient estimate at each iteration, multiple gradient estimates can be generated at additional computational cost and averaged to reduce the noise. An accelerated form of SPSA approximates the second-order Hessian $\nabla^2 f(x)$ to accelerate the convergence [40], analogous to the Newton–Raphson method. Iterate averaging in the SPSA setting has also been explored, but performs relatively poor in finite-time [13,39]. All in all, SPSA has been shown to be an effective SA method for tackling high-dimensional problems, with ease of implementation and the asymptotic theory to support it.

## 6.4 Recent Modifications

This section presents several recently proposed modifications that focus on improving the finite-time performance of SA: the scaled-and-shifted Kiefer–Wolfowitz (SSKW) algorithm, the robust SA (RSA) algorithm, the accelerated SA (AC-SA) algorithm, and the Secant-Tangents AveRaged stochastic approximation (STAR-SA) algorithm. The theoretical results for RSA and AC-SA focus on an alternative way to analyze the performance of the estimates through $f(x_N^*) - f(x^*)$. The inequality in (6.15) is an alternative way to view the performance of SA, which

focuses on the distance between the function evaluated at the estimate and the optimal function value (i.e., $\mathsf{E}[f(x_N^*) - f(x^*)]$) as opposed to a distance between the estimate and optimum (e.g., $\mathsf{E}(x_N^* - x^*)^2$). To illustrate the difference, consider an extremely flat function on the entire feasible region. The alternative performance measure will indicate that almost any iterate in the feasible region will be a good estimate, whereas performance based on the mean-squared error (MSE) of the estimate and optimum will be more sensitive to the estimate $x_N^*$. Further details on these two algorithms are provided in the next chapter.

### *6.4.1 Scaled-and-Shifted Kiefer–Wolfowitz (SSKW)*

The scaled-and-shifted Kiefer–Wolfowitz (SSKW) algorithm [4] adaptively adjusts $\{a_n\}$ and $\{c_n\}$ finitely many times during the course of the algorithm to adapt to the characteristics of the function and noise level in hopes of preventing slow convergence in finite-time. The idea is to increase $\{a_n\}$ so the iterates are able to make noticeable progress towards the optimum with the option of decreasing $\{a_n\}$ later if it is too large. Furthermore, if the direction of the gradient is classified as incorrect, then $\{c_n\}$ is increased to reduce the noise. Note that KW only requires two parameter choices $\{a_n\}$ and $\{c_n\}$, whereas SSKW requires eleven, as seen in the algorithm below.

---

**SSKW Algorithm**

**Scaling Phase**

- Input. $\{a_n\}$, $\{c_n\}$, $[l,u]$, $\Pi_\Theta$, stopping time $N$, and
  - $h_0$ = number of forced boundary hits,
  - $\gamma_0$ = scale up factor for $\{c_n\}$,
  - $k_a$ = maximum number of shifts of $\{a_n\}$,
  - $v_a$ = initial upper bound of shift,
  - $\phi_a$ = maximum scale up factor for $\{a_n\}$,
  - $k_c$ = maximum number of scale ups for $\{c_n\}$,
  - $c_0$ = maximum value of $\{c_n\}$ after scale ups (i.e., $c_n \leq c^{max} = c_0(u-l)$),
  - $g_0$ = maximum number of gradient estimates in scaling phase,
  - $m_{max}$ = maximum number of adaptive iterations ($m_{max} \leq N$).
- Initialize.
  - Choose $x_1 \in [l + c_1, u - c_1]$.
  - Let $n = 1$, $m = 1$, $g = 1$, $sh = 0$, and $sc = 0$.
- Do while $m \leq h_0$ and $g \leq g_0$.
  - Step 1.
    - Generate an estimate $\hat{\nabla} f(x_n)$ using symmetric differences.

- Compute $x_{n+1}$ using recursion (6.3).
  - If $x_{n+1} \in (l + c_n, x_n)$, go to Step 2.
  - If $x_{n+1} \in (x_n, u - c_n,)$, go to Step 3.
  - If $x_{n+1} > u - c_{n+1}$ and $x_n = u - c_n$ or if $x_{n+1} < l + c_{n+1}$ and $x_n = l + c_n$, go to Step 4, if $sc \leq k_c$.
  - If $x_{n+1} > u - c_{n+1}$ and $x_n = l + c_n$ or if $x_{n+1} < l + c_{n+1}$ and $x_n = u - c_n$, go to Step 5.

– Step 2.

- Scale $\{a_n\}$ up by $\alpha = \min\{\phi_a, (u - c_{n+1} - x_n)/(x_{n+1} - x_n)\}$ and use $\{\alpha a_n\}$ for the remaining iterations.
- Set $x_{n+1} = l + c_{n+1}$. Let $n = n + 1$, $m = m + 1$, $g = g + 1$ and go to Step 1.

– Step 3.

- Scale $\{a_n\}$ up by $\alpha = \min\{\phi_a, (l + c_{n+1} - x_n)/(x_{n+1} - x_n)\}$ and use $\{\alpha a_n\}$ for the remaining iterations.
- Set $x_{n+1} = u - c_{n+1}$. Let $n = n + 1$, $m = m + 1$, $g = g + 1$ and go to Step 1.

– Step 4.

- Scale $\{c_n\}$ up by $\gamma = \min\{\gamma_0, c^{max}/c_n\}$ and use $\{\gamma c_n\}$ for the remaining iterations.
- Let $sc = sc + 1$ and go to Step 5.

– Step 5.

- Set $x_{n+1} = \min\{u - c_{n+1}, \max\{x_{n+1}, l + c_{n+1}\}\}$.
- Let $n = n + 1$, $g = g + 1$ and go to Step 1.

**Shifting Phase**

- While $n \leq m_{max}$ and $n \leq N$,
  – Step 1.
  - Generate an estimate $\hat{\nabla} f(x_n)$ using symmetric differences.
  - Compute $x_{n+1}$ using (6.3).
    - If $x_{n+1} > u - c_{n+1}$ and $x_n = l + c_n$ or if $x_{n+1} < l + c_{n+1}$ and $x_n = u - c_n$, go to Step 2, if $sh < k_a$.
    - If $x_{n+1} > u - c_{n+1}$ and $x_n = u - c_n$ or if $x_{n+1} < l + c_{n+1}$ and $x_n = l + c_n$, go to Step 3, if $sc < k_c$.
    - Otherwise, go to Step 4.

  – Step 2.
  - Find smallest integer $\beta'$ such that $x_{n+1} \in (l + c_n, u - c_n)$ with $a_{n+\beta'}$.
  - Set $\beta = \min(v_a, \beta')$ and shift $\{a_n\}$ to $\{a_{n+\beta}\}$. If $\beta = v_a$, set $v_a = 2v_a$.
  - Let $sh = sh + 1$ and go to Step 4.

– Step 3.
  · Scale $\{c_n\}$ up by $\gamma = \min\{\gamma_0, c^{max}/c_n\}$ and use $\{\gamma c_n\}$ for the remaining iterations.
  · Let $sc = sc + 1$ and go to Step 4.
– Step 4.
  · Set $x_{n+1} = \min\{u - c_{n+1}, \max\{x_{n+1}, l + c_{n+1}\}\}$.
  · Let $n = n + 1$ and go to Step 1.

**KW Algorithm**

- If $n > m_{max}$ and $n < N$, then SSKW reverts back to KW and stop when $n = N$.
- Output. $x_N^* = x_N$.

---

The SSKW algorithm has two pre-processing phases, scaling and shifting, which adjust the tuning sequences in order to improve the finite-time performance, before reverting back to the original KW algorithm. In the scaling phase, the $\{a_n\}$ is scaled up by a factor $\alpha$, i.e., $\{a_n\}$ to $\{\alpha a_n\}$, so the iterates can move from one boundary to the other to ensure the step sizes are not too small relative to the gradient. In the shifting phase, the sequence $\{a_n\}$ is decreased by shifting or "skipping" a finite number $(\beta)$ of terms from $\{a_n\}$ to $\{a_{n+\beta}\}$, when the iterates fall outside of the feasible region when the sign of the gradient is correct. This acts as a recourse stage and reduces the step size faster in case the step-size sequence $\{a_n\}$ is too large. During both phases, $\{c_n\}$ is scaled up by $\gamma$, i.e., $\{c_n\}$ to $\{\gamma c_n\}$, if the previous iterate is at the boundary and the update falls outside the feasible region but is moving in the wrong direction. This increase is an attempt to reduce the noise of the gradient estimate. These adjustments do not affect the asymptotic convergence, since the scaling phase only scales $\{a_n\}$ up by a constant, the shifting phase only skips a finite number of terms in $\{a_n\}$, and the perturbation sequence $\{c_n\}$ is only scaled up by a constant, all of which occur finitely many times.

### *6.4.2 Robust Stochastic Approximation (RSA)*

The robust SA (RSA) method is intended to be relatively insensitive to the choice of the step-size sequence, similar to Polyak–Ruppert iterate averaging. The form of RSA is identical to (6.3) with the exception of the output. Instead of $x_N^* = x_N$, where $x_N$ is the last iterate, $x_N^*$ is calculated as

$$x_N^* = \frac{\sum_{n=1}^{N} a_n x_n}{\sum_{n=1}^{N} a_n},$$

where $a_n > 0$ for all $n$. It is clear that if $a_n = a$, where $a \in \mathbb{R}^+$ for all $n$, then $x_N^* = \frac{1}{N}\sum_{n=1}^N x_n$, giving the uniformly weighted average of Polyak–Ruppert. As mentioned earlier, iterate averaging under a constant step size for a moving window is asymptotically normal [28]. A finite-time bound was derived for $\mathsf{E}[f(x_n^*) - f(x^*)]$ under RSA when $f$ is assumed convex [30]. Assume there exists $C > 0$ such that $\mathsf{E}[||\nabla f(x)||^2] \leq C^2$ for all $x \in \Theta$. Then for an $N$-step iteration policy,

$$\mathsf{E}[f(x_N^*) - f(x^*)] \leq \frac{||x_0 - x^*||^2 + C^2 \sum_{n=1}^N a_n^2}{2\sum_{n=1}^N a_n}. \tag{6.8}$$

For equal weights or iterate averaging, the bound on the right hand side of (6.8) can be minimized if

$$a_n = a := \frac{D_\Theta}{C\sqrt{N}},$$

where $D_\Theta = \max_{x,y\in\Theta} ||x - y||$. The distance $||x_0 - x^*||$ in the place of $D_\Theta$ tightens the bound in (6.8), but $x^*$ is unknown so the improvement may not be practically meaningful. This step size requires the number of iterations $N$ to be fixed. Similar to iterate averaging, a sliding window average can also be employed in RSA. The estimate consists of the last $N - K + 1$ estimates and has the form

$$x_{N,K}^* = \frac{\sum_{n=K}^N a_n x_n}{\sum_{n=K}^N a_n}. \tag{6.9}$$

If we consider the varying step size

$$a_n = \frac{\theta D_\Theta}{C\sqrt{n}}, \tag{6.10}$$

for $\theta > 0$, then we have the bound

$$\mathsf{E}[f(x_{N,K}^*) - f(x^*)] \leq \frac{D_\Theta C}{\sqrt{N}}\left[\frac{2}{\theta}\left(\frac{N}{N-K+1}\right) + \frac{\theta}{2}\sqrt{\frac{N}{K}}\right], \tag{6.11}$$

for $1 \leq K \leq N$.

### 6.4.3 *Accelerated Stochastic Approximation (AC-SA) for Strongly Convex Functions*

The accelerated SA (AC-SA) algorithm [21] takes a similar approach to iterate averaging and RSA by taking long strides and incorporating each of the iterates into

the output. The next two algorithms, accelerated SA for strongly convex and convex functions, take advantage of the smoothness factor of the function if it exists. AC-SA for convex functions is a special case of AC-SA for strongly convex functions, so we first introduce AC-SA for strongly convex functions and then restrict the strong convexity parameter for the convex case.

AC-SA is an example of a proximal method, which introduce a proximity function into the objective function. The prox-function acts as a regularization term to prevent the next iterate update $x_{n+1}$ from being too far from $x_n$ and is comprised of a distance generating function or Bregman function $\omega : \Theta \to \mathbb{R}$, which is continuously differentiable and strongly convex with modulus $\nu > 0$ satisfying

$$\langle x - y, \nabla\omega(x) - \nabla\omega(y)\rangle \geq \nu||x-y||^2 \ \forall x, y \in \Theta,$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product. A prox-function with the given distance generating function is

$$V(x,y) = V_\omega(x,y) = \omega(y) - [\omega(x) + \langle\nabla\omega(x), y - x\rangle].$$

As $x_n \to x^*$, the regularization term disappears, so minimizing $f(x)$ plus a regularizer is equivalent to minimizing the function $f(x)$.

Consider a strongly convex function $f(\cdot)$ satisfying

$$\frac{\mu}{2}||y-x||^2 \leq f(y) - f(x) - \langle\nabla f(x), y - x\rangle \leq \frac{L}{2}||y-x||^2 + M||y-x||, \quad (6.12)$$

for all $x, y \in \Theta$ where $\mu > 0$ is the strong convexity parameter. Notice that if $f$ is Lipschitz continuous with Lipschitz constant $M/2$, then (6.12) holds with $M > 0$, $L = 0$, and $\mu = 0$, and if $f$ has Lipschitz continuous gradients with Lipschitz constant $L$, then (6.12) holds with $M = 0$, $L > 0$, and $\mu = 0$.

The AC-SA algorithm updates three sequences, $\{x_n^{md}\}, \{x_n^{ag}\}$, and $\{x_n\}$. Here, "*md*" and "*ag*" are abbreviations for median and aggregate, respectively, and median is used in a loose sense.

---

**Accelerated SA Method for Strongly Convex Functions**

- Input.
  - Specify $V(x,y)$, $\{\alpha_n\}$ and $\{\gamma_n\}$ be given such that $\alpha_1 = 1$, $\alpha_n \in (0,1)$ for $n \geq 2$, and $\gamma_n > 0$ for $n \geq 1$ and a stopping time $N$.
- Initialize. Choose $x_0^{ag} = x_0 \in \Theta$ and let $n = 1$.
- While $n < N$,
  - Step 1. Generate an estimate $\hat{\nabla} f(x_n)$ of $\nabla f(x_n)$.
  - Step 2. Compute

$$x_n^{md} = \frac{\alpha_n[(1-\alpha_n)\mu+\gamma_n]}{\gamma_n+(1-\alpha_n^2)\mu}x_{n-1} + (1-\alpha_n)\frac{(1-\alpha_n)(\mu+\gamma_n)}{\gamma_n+(1-\alpha_n^2)\mu}x_{n-1}^{ag}$$

$$x_n = \arg\min_{x\in\Theta}\{\alpha_n[\langle\nabla f(x_n^{md}),x\rangle + \mu V(x_n^{md},x)] + [(1-\alpha_n)\mu+\gamma_n]V(x_{n-1},x)\}$$

$$x_n^{ag} = \alpha_n x_n + (1-\alpha_n)x_{n-1}^{ag}$$

– Step 3. Let $n = n+1$ and go to Step 1.

- Output. $x_N^* = x_N^{ag}$.

Note: $V(x,y) = \frac{1}{2}||x-y||^2$ using the Euclidean norm with $\nu = 1$ is a common prox-function. Refer to [20] for details.

---

**Theorem 6.5 (Theorem 1 [20]).** *Assume* $V(x,y) \le \frac{1}{2}||x-y||^2$ for all $x,y\in\Theta$ *when* $\mu < 0$ *and* $\mathsf{E}[(\hat{\nabla}f(x)-\nabla f(x))^2] \le \sigma^2\ \forall x\in\Theta$. *Choose* $\{\alpha_n\}$ *and* $\{\gamma_n\}$ *such that*

$$\nu(\mu+\gamma_n) > L\alpha_n^2, \tag{6.13}$$

$$\gamma_n/\Gamma_n = \gamma_{n+1}/\Gamma_{n+1} \quad \text{for } n \ge 1, \tag{6.14}$$

*where*

$$\Gamma_n = \begin{cases} 1 & \text{if } n = 1; \\ (1-\alpha_n)\Gamma_{n-1} & \text{if } n \ge 2. \end{cases}$$

*Then,*

$$\mathsf{E}[f(x_N^{ag}) - f(x^*)] \le \Gamma_N\left(\gamma_1 V(x_0,x^*) + \sum_{n=1}^{N}\frac{2(M^2+\sigma^2)\alpha_n^2}{\Gamma_n[\nu(\mu+\gamma_n)-L\alpha_n^2]}\right). \tag{6.15}$$

Consider $\alpha_n = 2/(n+1)$, $\gamma_n = 4L/[\nu n(n+1)]$, and $\Gamma_n = 2/[n(n+1)]$. It can be easily checked that these choices satisfy conditions (6.13) and (6.14). Under these conditions, the right hand side of (6.15) can be bounded by

$$\frac{4LV(x_0,x^*)}{\nu N(N+1)} + \frac{8(M^2+\sigma^2)}{\nu\mu(N+1)}, \tag{6.16}$$

for $\mu > 0$. The bounds in (6.15) and (6.16) rely on additional information of the function and gradient, which are unknown, so they must be approximated.

### 6.4.4 Accelerated Stochastic Approximation (AC-SA) for Convex Functions

AC-SA for convex functions is a special case of AC-SA for strongly convex functions with $\mu = 0$. The algorithm is identical to AC-SA for strongly convex function with the exception of the $x_n^{md}$ and $x_n$ update since $\mu = 0$. The resulting updates are

$$\begin{aligned} x_n^{md} &= \alpha_n x_{n-1} + (1-\alpha_n) x_{n-1}^{ag}, \\ x_n &= \arg\min_{x\in\Theta} \{\alpha_n \langle \nabla f(x_n^{md}), x\rangle + \gamma_n V(x_{n-1}, x)\}. \end{aligned}$$

Interestingly, if $V(x,y) = \frac{1}{2}||x-y||^2$, then the update for $x_n$ simplifies to

$$x_n = \Pi_\Theta \left( x_{n-1} - \frac{\alpha_n}{\gamma_n} \hat{\nabla} f(x_n^{md}) \right), \tag{6.17}$$

which has a similar form to the standard SA algorithm. Notice in the update for $x_n$ in (6.17), $\alpha_n/\gamma_n$ takes the place of the step size $a_n$ in (6.3) and the gradient estimate $\hat{\nabla} f$ is evaluated at $x_n^{md}$ as opposed to $x_{n-1}$. If we consider the same parameter setting as in the strongly convex case, the "step size" $\alpha_n/\gamma_n$ increases with $n$. Furthermore, the lower and upper bounds for the optimal objective function can be computed online and the difference converges to 0 as the number of iterations goes to infinity [20].

**Theorem 6.6 (Proposition 7 [20]).** *Assume that the assumptions in Theorem 6.5 hold for $\mu = 0$ and the sequences $\alpha_n = 2/(n+1)$ and $\gamma_n = 4\gamma/[\nu n(n+1)]$ for $\gamma \geq 2L$. Then*

$$\mathsf{E}[f(x_N^{ag}) - f(x^*)] \leq \frac{4\gamma V(x_0, x^*)}{\nu N(N+1)} + \frac{4(M^2+\sigma^2)(N+2)}{3\gamma}, \tag{6.18}$$

*where*

$$\gamma = \max\left\{ 2L, \left[ \frac{\nu(M^2+\sigma^2)N(N+1)(N+2)}{3V(x_0,x^*)} \right]^{1/2} \right\}$$

*minimizes the bound in (6.18).*

### 6.4.5 Secant-Tangents AveRaged Stochastic Approximation (STAR-SA)

The Secant-Tangents AveRaged (STAR) stochastic approximation algorithm estimates the gradient using a hybrid estimator, which is a convex combination of a symmetric finite difference and an average of two direct gradient estimators:

$$\hat{\nabla} f(x_n) = \alpha_n \frac{Y(x_n + c_n, \varepsilon_n^+) - Y(x_n - c_n, \varepsilon_n^-)}{2c_n} + (1-\alpha_n)\left(\frac{Y'(x_n + c_n, \delta_n^+) + Y'(x_n - c_n, \delta_n^-)}{2}\right), \tag{6.19}$$

where $\varepsilon_n^{\pm}$ and $\delta_n^{\pm}$ denote the randomness (i.e., $f(x_n \pm c_n) = \mathsf{E}[Y(x_n \pm c_n, \varepsilon_n^{\pm})]$ and $f'(x_n \pm c_n) = \mathsf{E}[Y'(x_n \pm c_n, \xi_n^{\pm})]$), $\alpha_n \in [0,1]$ for all $n$, $c_n \to 0$ and $\alpha_n \to 0$ as $n \to \infty$. The STAR gradient estimate requires function and gradient estimates on two points, $x_n \pm c_n$ for each $\hat{\nabla} f(x_n)$. In a setting where direct gradients are available, if the direct gradient is very noisy relative to the function estimates, it is difficult to decide between implementing RM or KW, even though RM converges faster asymptotically. Since the performance of neither algorithm is always superior to the other, the STAR gradient incorporates both. The weights of the convex combination play a critical role in the performance of STAR-SA and can be chosen to minimize the variance of the gradient estimate such that it is less than the variance of both the symmetric finite difference gradient estimate and direct gradient estimate. If

$$\alpha_n^* = \frac{\sigma_g^2 c_n^2 + \rho \sigma_f \sigma_g c_n^2}{\sigma_f^2 + \sigma_g^2 c_n^2 + 2\rho \sigma_f \sigma_g c_n},$$

where $\mathrm{Var}[Y(x,\varepsilon)] = \sigma_f^2$, $\mathrm{Var}[Y'(x,\xi)] = \sigma_g^2$, and $\mathrm{Corr}(Y(x,\varepsilon), Y'(x,\xi)) = \rho$, then STAR-SA is theoretically optimal in terms of MSE compared to RM and KW for simple quadratic functions, and the variance of the STAR gradient is less than that of RM and KW under certain conditions.

**Theorem 6.7 (Theorem 3 [6]).** *Let $\{x_n\}$ be a sequence generated using recursion (6.3) and gradient estimate (6.19). Assume*

1. *There exist positive sequences $\{a_n\}$, $\{c_n\}$, and $\{\alpha_n\}$ such that $\alpha_n \in [0,1]$ for all $n$, $\sum_{n=1}^{\infty} a_n \alpha_n = \infty$, $\sum_{n=1}^{\infty} a_n c_n < \infty$, $\sum_{n=1}^{\infty} a_n^2 < \infty$, and $\sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty$.*
2. *There exist $B, C > 0$ such that $\mathsf{P}(|f''(x)| \le B) = 1$ and $\mathsf{P}(|f'(x)| \le C) = 1$ for all $x \in \Theta$.*
3. *There exist $K_0, K_1 > 0$ such that $K_0|x - x^*| \le |f'(x)| \le K_1|x - x^*|$ for all $x \in \Theta$.*
4. *$f'(x)(x - x^*) > 0$ for all $x \in \mathbb{R} \backslash \{x^*\}$.*
5. *For $c > 0$, $\sigma^2 = \sup_{x \in \mathbb{R}} \mathrm{Var}[Y(x + c, \xi^+) - Y(x - c, \xi^-)|x] < \infty$ for all $x \in \Theta$.*
6. *$\varepsilon_n^+$, $\varepsilon_n^-$, $\delta_n^+$, $\delta_n^-$ are i.i.d. with mean zero for all $n$.*

*Then $x_n \xrightarrow{L^2} x^*$ as $n \to \infty$.*

Numerical experiments show that STAR-SA is competitive against RM and KW, even when the number of iterations for RM are doubled due to the increase in computational cost of STAR-SA [6]. In the experimental results, the STAR-SA algorithm either performs significantly better than RM and KW or the MSE is close to that of the algorithm with the lower MSE. STAR-SA has been extended to higher dimensions by considering simultaneous perturbation gradient estimates, instead of using a symmetric finite difference gradient estimate, to take advantage of its potential efficiency and robustness [5].

## 6.5 Numerical Experiments

We present three sets of numerical experiments comparing the mean-squared error (MSE) of various SA algorithms on several contrasting functions. The first set of experiments illustrates the sensitivity of KW and two variants to the choice of the two step-size sequence parameters, taken from [7]. The second set compares three SA algorithms, the robust SA (RSA) method, the accelerated SA (AC-SA) method, and the original RM algorithm, under various initial settings and step-size parameters for RM (i.e., starting values, compact intervals, noise levels, and step sizes). The last set of numerical experiments explores the potential gains from using the STAR gradient estimate, which utilizes both direct and indirect gradient estimates, as opposed to using them separately, as in RM and KW, respectively. Since the numerical experiments consider maximization problems, the sign of $a_n$ and $\alpha_n/\gamma_n$ in recursion (6.3) and (6.17), respectively, must be adjusted accordingly.

### *Sensitivity Analysis of KW and Its Variants*

We perform a sensitivity analysis of KW and KW using Kesten's rule (denoted henceforth by KWK) with symmetric finite difference gradient estimates, and we compare the results with SSKW. Using the parameter settings $a_n = \theta_a/n$, $c_n = \theta_c/n^{1/4}$, $\theta_a > 0$, $\theta_c > 0$ arbitrary but fixed, $N = 10{,}000$ iterations, and 1,000 sample paths, our analysis replicates the results of [4] for $f(x) = -0.001x^2$ on the interval $[-50, 50]$, where SSKW performs significantly better than KW in terms of MSE and oscillatory period; however, this result is obtained using what seem to be nearly worst-case parameter setting for KW. In our experiments, we consider a wide range of parameters and initial settings for KW and KWK: 19 initial starting values uniformly spaced within the truncated interval $x_1 \in \{-50+5k \mid k = 1,2,\ldots 19\}$, 45 different $\theta_a$ values parametrized by $\theta_a \in \{10^s k \mid k = 1,2,\ldots,9, s = 0,1,\ldots,4\}$, and 10 different $\theta_c$ values parametrized by $\theta_c \in \{10^s k \mid k = 1,2,\ldots,5, s = 0,1\}$. In total, there are 8,550 combinations.

The numerical results illustrate the sensitivity of the classical SA methods to the parameters. In fact, near optimal performance can be obtained with fine-tuning.

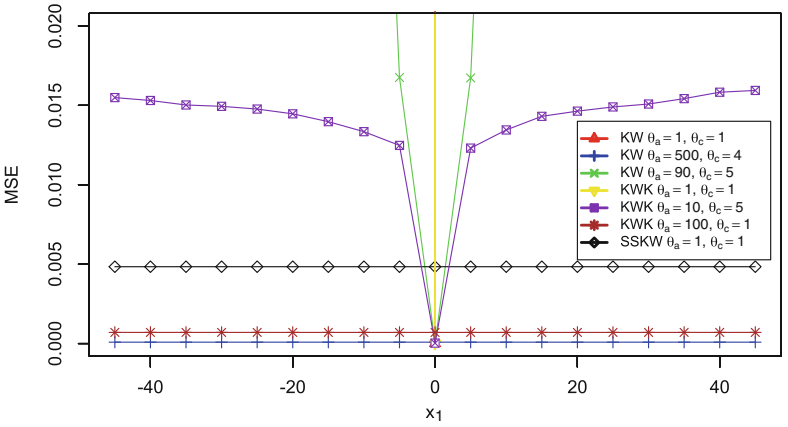


**Fig. 6.2** MSE of the 10000th iterate of KW and KWK for three parameter settings and SSKW for $f(x) = -.001x^2$, $\sigma = 0.001$, $a_n = \theta_a/n$, $c_n = \theta_c/n^{1/4}$

Out of the 8,550 combinations, KW outperforms SSKW in half of the cases, which indicates that with some tuning, KW yields good performance for a fairly wide range of tunable parameters. Figure 6.2 plots the MSE of KW, KWK, and SSKW for $f(x) = -.001x^2, \sigma = 0.001$ against the initial starting values $x_1$ for several parameter choices that are a good representation of a majority of the results. The parameter value $\theta_a = \theta_c = 1$, identical to the settings in [4], is among the worst for KW and KWK, represented by a nearly vertical orange line for both algorithms, as a result of the overlapping red and yellow lines for KW and KWK, respectively. For this parameter setting, SSKW beats KW and KWK significantly for all initial values with the exception of $x_1 = 0$. The first column in Table 6.1 compares the MSE all three algorithms with $x_1 = 0.01$, and clearly, KW outperforms SSKW in almost all cases. Of course, a practitioner would have no way of knowing whether or not the starting iterate was close to the true optimum, so these results do not indicate that KW will always perform well. They do indicate, however, that KW exhibits substantial variation in performance. In the case where $\theta_a = 90, \theta_c = 5$ and $\theta_a = 10, \theta_c = 5$, KW and KWK, respectively, outperform SSKW in a neighborhood around the optimum. There are also well-tuned parameters such as $\theta_a = 500,\ \theta_c = 4$ for KW and $\theta_a = 100,\ \theta_c = 1$ for KWK that outperform SSKW for all initial start values. When KW and KWK perform better than SSKW, the difference is not as pronounced as when SSKW outperforms KW, but careful tuning can partially mitigate the sensitivity of KW to parameters such as the initial iterate.

In addition, we implement KW and its variants using the same parameters ($a_n = 1/n,\ c_n = 1/n^{1/4},\ x_1 = 30$) as in [4] on $f(x) = 100e^{-.006x^2}$ to test the algorithms under the same setting for a different function. Figure 6.3 plots the MSE of the 10000th iterate as a function of the initial start value. The horizontal line for all noise levels indicates that SSKW is insensitive the initial start value. KW and KWK outperform SSKW within certain intervals around the optimum for

**Table 6.1** MSE of the 100th, 1000th, and 10000th iteration for KW and its variants with $a_n = 1/n$, $c_n = 1/n^{1/4}$

| | | $f(x) = -0.001x^2$ [-50, 50] $x_1 = .01$ | | | $f(x) = 100e^{-0.006x^2}$ [-50, 50] $x_1 = 30$ | | |
|---|---|---|---|---|---|---|---|
| $\sigma$ | Algorithm | 100 | 1000 | 10000 | 100 | 1000 | 10000 |
| 0.001 | SSKW | $5.10x10^{-2}$ | $1.70x10^{-2}$ | $5.00x10^{-3}$ | $5.07x10^{-2}$ | $1.68x10^{-2}$ | $4.84x10^{-3}$ |
| | KW | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | 763.8 | 653.3 | 431.4 |
| | KWK | $1.12x10^{-4}$ | $1.08x10^{-4}$ | $1.04x10^{-4}$ | $10^{-7}$ | $3x10^{-8}$ | $10^{-8}$ |
| 0.01 | SSKW | $5.10x10^{-2}$ | $1.70x10^{-2}$ | $5.00x10^{-3}$ | 5.07 | 1.68 | $4.90x10^{-1}$ |
| | KW | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | 763.8 | 653.3 | 431.2 |
| | KWK | $2.10x10^{-3}$ | $2.11x10^{-3}$ | $2.05x10^{-3}$ | $9.54x10^{-6}$ | $2.76x10^{-6}$ | $8.41x10^{-7}$ |
| 0.1 | SSKW | $5.10x10^{-2}$ | $1.70x10^{-2}$ | $5.00x10^{-3}$ | 165.8 | 57.4 | 16.0 |
| | KW | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | 763.4 | 651.4 | 418.2 |
| | KWK | $2.01x10^{-1}$ | $2.03x10^{-1}$ | $1.97x10^{-1}$ | $5.65x10^{-2}$ | $2.76x10^{-4}$ | $8.41x10^{-5}$ |
| 1.0 | SSKW | $5.10x10^{-2}$ | $1.70x10^{-2}$ | $5.00x10^{-3}$ | 187.2 | 57.8 | 18.7 |
| | KW | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | 722.5 | 562.5 | 415.7 |
| | KWK | 20.1 | 20.3 | 19.7 | 456.9 | 315.1 | 239.7 |

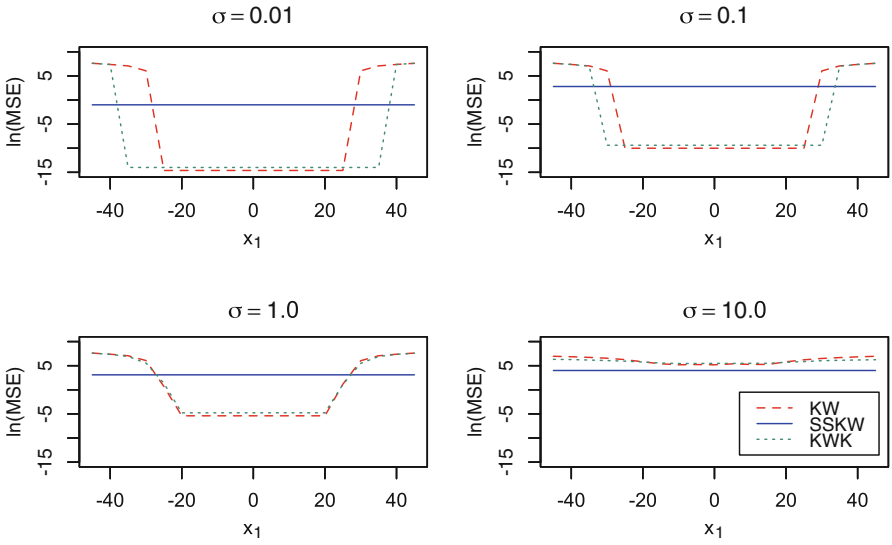


**Fig. 6.3** MSE Comparison of KW and its variants for $f(x) = 100e^{-.006x^2}$ with values $a_n = 1/n$, $c_n = 1/n^{1/4}$, $N = 10000$

$\sigma \in \{0.001, 0.01, 0.1, 1.0\}$ and KWK's better performance intervals overlap those of KW. However, KW using the deterministic step size $1/n$ performs better than KWK where the intervals overlap, which can be seen in Fig. 6.3. Unfortunately, outside of those intervals, KW and KWK have a tendency to perform poorly.

### *RM, RM with Iterate Averaging, Robust SA and Accelerated SA*

We investigate the MSE performance of RSA and AC-SA using direct gradient estimates and compare the results against the classical RM algorithm and RM with iterate averaging. We consider the optimal parameter settings for RSA and AC-SA, which require additional knowledge of the function, its gradient, and the optimum, so in practice, they must be approximated.

We consider a simple quadratic function, $f(x) = -\frac{1}{3}x^2$, on the truncated intervals $[-50, 50]$ and $[-5, 95]$ with $x_1 = 30.0$, $\sigma = 1.0$, and 1,000 sample paths. For the RM and RM with iterate averaging algorithm, we employ a common step size $a_n = \theta_a/n$, where $\theta_a = 10.0$. RM performed relatively well for a wide range of multiplicative constants. We chose to use $\theta_a = 10.0$, although it did not yield the lowest MSE at the 1000th or 10000th iteration from preliminary numerical tests. For RSA, we adopt a constant step size that minimizes the finite-time bound in (6.8), where $C = 100/3, 190/3$ for the intervals [-50, 50] and [-5, 95], respectively, and $D_\Theta = 100$. For the AC-SA algorithm, we consider $\alpha_n = 2/(n+1)$ and $\gamma_n = 4\gamma/[n(n+1)]$, where $\gamma$ is given in (6.19) with $\nu = 1$, $L = 2/3$ and $M = 0$.

Figure 6.4 plots the MSE as a function of the number of iterations from 1 to 10000 on a log scale. The results for both the centered and skew truncated intervals appear to have the same behavior across all four algorithms. RM performs well with a good parameter choice, although it is not the best, but averaging the iterates improves the performance, resulting in a smoother monotonically decreasing MSE curve as the number of iterations increase. Compared to a decently/reasonably tuned RM and RM with iterate averaging algorithm, RSA appears to be inferior, at least in this simple numerical experiment. The most interesting curve is from the AC-SA algorithm, where one can observe periodic oscillations, which decrease in magnitude as the number of iterations increase. We further investigated this behavior by analyzing individual sample paths, and the estimates $\{x_n^{ag}\}$ appear to have the same behavior, following a smooth oscillating path/curve. From Fig. 6.4, the AC-SA curve appears to level off and hover slightly over the RSA curve. The stopping time dictates the relative performance of AC-SA when there are a smaller number of iterations because of the oscillations. For the case of the skewed interval, there is a small range of iterations where AC-SA outperforms RSA, RM, and RM with iterate averaging, as well as other small ranges where it outperforms RSA. Keep in mind that these experiments are for a simple quadratic function for a particular setting, so the relative performance will most likely change in a different setting.

From our numerical experiments, one can conclude that RM and RM with iterate averaging has the potential to outperform RSA and AC-SA if the step-size parameter is chosen appropriately for a wide range of choices. In this case, iterate averaging improves the performance of RM for all 10,000 iterations. Both the AC-SA and RSA algorithms require additional knowledge to choose the optimal step size that minimizes the bound in (6.15) and (6.8) for AC-SA and RSA, respectively.

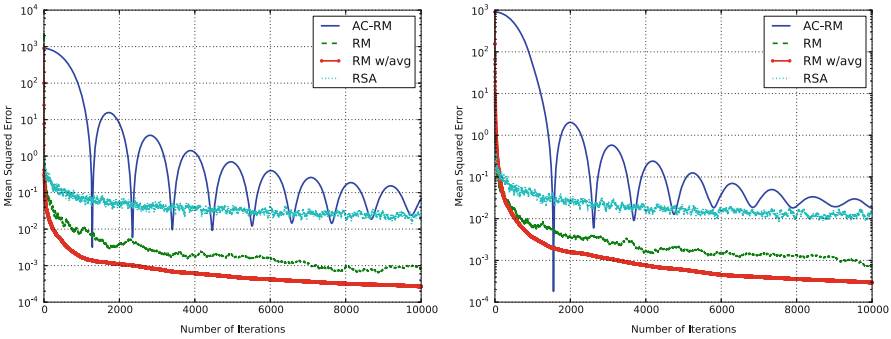


**Fig. 6.4** MSE under RM, RM with averaging, RSA, and AC-SA for $f(x) = -\frac{1}{3}x^2$, $x_1 = 30.0$, $\sigma = 1.0$, for symmetrical $[-50, 50]$ (*left graph*) and skewed $[-5, 95]$ (*right graph*) intervals

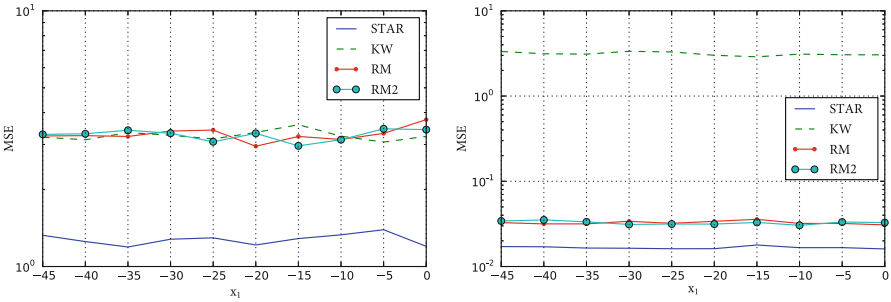


**Fig. 6.5** MSE of 1000th iterate under STAR-SA, RM, and KW for $f(x) = -0.1x^2$, $\sigma_g = 1.0$, and two levels of $\sigma_f$: 0.1 (*left graph*) and 1.0 (*right graph*)

## *STAR-SA, RM, and KW*

We implement STAR-SA, RM, and KW under various combinations of noise levels $\sigma_f$ and $\sigma_g$, for $f(x) = -ax^2$, $a > 0$ and $\Theta = [-50, 50]$. The gain sequence and finite difference step sizes are $\theta_a/n$ and $\theta_c/n^{1/4}$, respectively, and the MSE results are based on 1,000 sample paths. For a fairer comparison, the number of iterations for RM is doubled, since STAR-SA and KW require twice the number of sample path runs. We consider the following values for the parameter and initial settings: steepness level $a \in \{10^k | k = -3, -2.5, \ldots, 1.5, 2\}$, $x_1 \in \{-50 + 5k | k = 1, \ldots, 10\}$, $\theta_a \in \{1, 10, 100\}$, $\theta_c \in \{0.1, 1.0\}$, $\sigma_f \in \{10^k | k = -3, \ldots, 1\}$, $\sigma_g \in \{10^k | k = -3, \ldots, 1\}$, and $N \in \{100, 1000, 10000\}$. Although STAR-SA, RM, and KW were implemented for all settings, only the case $f(x) = -0.1x^2$, $a_n = 10n^{-1}$, $c_n = 0.1n^{-1/4}$, $\sigma_f \in \{0.001, 0.1, 1.0\}$, $\sigma_g \in \{0.001, 0.1, 1.0\}$ and $N = 1000$ will be described in detail.

The STAR-SA algorithm outperforms KW and RM for 6 out of the 9 combinations for all initial start values considered. For $\sigma_f = 0.001$ and $\sigma_f < \sigma_g$, the MSE of STAR-SA is lower than that of RM, but is approximately equal

to that of KW. The only case where the MSE of the STAR-SA algorithm is not approximately less than or equal to the MSE of KW and RM is when both noise levels are very low, i.e., $\sigma_f = \sigma_g = 0.001$, which is not shown. In this case, RM performs better than STAR-SA except when the start value is close to the optimum $x^* = 0$. In fact, the MSE of STAR-SA decreases as $x_1$ approaches $x^*$. In addition, the MSE of KW and RM are close to the optimum when $\sigma_f = 0.001$ and $\sigma_g = 0.001, 0.1$, respectively, whereas the MSE of STAR-SA is close to 0 for $(\sigma_f, \sigma_g) \in \{(0.001, 0.001), (0.001, 0.1), (0.001, 1.0), (0.1, 0.001), (0.1, 0.1), (1.0, 0.001), (1.0, 0.1)\}$. Figure 6.5 illustrates the MSE results when $\sigma_f = 1.0$. When the noise of the function is high, KW performs poorly, RM outperforms KW, and STAR-SA has the lowest MSE. Figure 6.5 shows a case where the performance of KW and RM are similar, but the MSE of the STAR-SA algorithm is lower. Overall, from the numerical experiments conducted, STAR-SA either performs significantly better than both RM and KW in terms of MSE or the MSE is approximately equal to that of the algorithm with the lower MSE.

## 6.6 Concluding Remarks

Stochastic approximation has an enormous body of literature in all aspects of theory, algorithms, and applications. From its origins in statistics, it has now reached many disciplines in engineering and the social sciences, with well-known successes in such areas as signal processing, pattern recognition, and machine learning. Clearly, simulation optimization is another fertile area for its application.

This chapter introduced the two main versions of SA:

- KW-like methods that rely only on function estimates, known as *gradient-free* or *stochastic zeroth-order* algorithms; and
- RM-like methods that make use of direct estimates of first-order derivative information, known as *stochastic gradient* or *stochastic first-order* algorithms.

The latter methods generally perform better in practice, but they require information that is not always available. Asymptotically, they can obtain an $O(n^{-1/2})$ convergence rate, whereas the former are generally limited to a $O(n^{-1/3})$ convergence rate. Among the gradient-free methods, SPSA has been particularly successful for high-dimensional problems.

The finite-time behavior of any SA algorithm depends heavily on the choice of the step-size or gain sequence, and various approaches to handling this challenge have been presented, from Kesten's rule to iterate averaging, with the latter procedure highly recommended.

Classical notions of convergence in SA address the iterates $\{x_n\}$, whereas recent finite-time analysis has turned to the properties of the function values $\{f(x_n)\}$. Chapter 7 focuses on some recent SA algorithms mainly tailored to convex stochastic programming problems and provides such convergence properties.

Finally, SA methods are aimed at continuous-valued optimization problems, but there is some work attempting to apply SA to discrete optimization problems. A recent Ph.D. dissertation [44] addresses this setting and includes a summary of previous work in the area.

**Acknowledgements** This work was supported in part by the National Science Foundation under Grants CMMI 0856256 and ECCS 0901543, and by the Air Force Office of Scientific Research under Grant FA9550-10-10340.

## References

1. S. Andradóttir. A stochastic approximation algorithm with varying bounds. *Operations Research*, 43(6):1037–1048, 1995.
2. J. R. Blum. Multidimensional stochastic approximation methods. *The Annals of Mathematical Statistics*, 25(4):737–744–200, 1954.
3. M. Broadie, D. Cicek, and A. Zeevi. An adaptive multidimensional version of the Kiefer-Wolfowitz stochastic approximation algorithm. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, editors, *Proceedings of the Winter Simulation Conference*, pages 601–612. IEEE, Piscataway, NJ, 2009.
4. M. Broadie, D. Cicek, and A. Zeevi. General bounds and finite-time improvement for the Kiefer-Wolfowitz stochastic approximation algorithm. *Operations Research*, 59(5):1211–1224, 2011.
5. M. Chau, M. C. Fu, and H. Qu. Multivariate stochastic approximation using a Secant-Tangents AveRaged (STAR) gradient estimator. Technical report, Working paper, University of Maryland, College Park, 2014.
6. M. Chau, H. Qu, and M. C. Fu. A new hybrid stochastic approximation algorithm. In *Proceedings of the 12th International Workshop on Discrete Event Systems*, 2014.
7. M. Chau, H. Qu, M. C. Fu, and I. O. Ryzhov. An empirical sensitivity analysis of the Kiefer-Wolfowitz algorithm and its variants. In R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, editors, *Proceedings of the 2013 Winter Simulation Conference*, pages 945–965. IEEE, Piscataway, NJ, 2013.
8. H. F. Chen. Convergence rate of stochastic approximation algorithms in the degenerate case. *SIAM Journal on Control and Optimization*, 36(1):100–114, 1998.
9. H. F. Chen, T. E. Duncan, and B. Pasik-Duncan. A Kiefer-Wolfowitz algorithm with randomized differences. *IEEE Transactions on Automatic Control*, 44(3):442–453, 1999.
10. H. F. Chen and Y. M. Zhu. Stochastic approximation procedure with randomly varying truncations. *Scientia Sinica Series A*, 29:914–926, 1986.
11. B. Delyon and A. B. Juditsky. Accelerated stochastic approximation. *SIAM Journal on Optimization*, 3(4):868–881, 1993.
12. C. Derman. An application of Chung's lemma to the Kiefer-Wolfowitz stochastic approximation procedure. *The Annals of Mathematical Statistics*, 27(2):532–536, 1956.
13. J. Dippon and J. Renz. Weighted means in stochastic approximation of minima. *SIAM Journal on Control Optimization*, 35(5):1811–1827, 1997.
14. V. Dupač. On the Kiefer-Wolfowitz approximation method. *Časopis pro péstování Matematiky*, 82(1):47–75, 1957.
15. V. Fabian. Stochastic approximation of minima with improved asymptotic speed. *The Annals of Mathematical Statistics*, 38(1):191–200, 1967.
16. V. Fabian. Stochastic approximation. In Rustagi, editor, *Optimizing Methods in Statistics*, pages 439–470. Academic Press, 1997.

17. M. C. Fu and S. D. Hill. Optimization of discrete event systems via simultaneous perturbation stochastic approximation. *IIE Transactions*, 29(3):233–243, 1997.
18. A. P. George and W. B. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65(1):167–198, 2006.
19. L. Gerencsér. Convergence rates of moments in stochastic approximation with simultaneous perturbation gradient approximation and resetting. *IEEE Transactions on Automatic Control*, 44(5):894–905, 1998.
20. S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization I: a generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.
21. S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization II: shrinking procedures and optimal algorithms. *SIAM Journal on Optimization*, 23(4):2061–2089, 2013.
22. C. Kao, W. T. Song, and S. P. Chen. A modified quasi-Newton method for optimization in simulation. *International Transactions in Operations Research*, 4(3):223–233, 1997.
23. H. Kesten. Accelerated stochastic approximation. *The Annals of Mathematical Statistics*, 29(1):41–59, 1958.
24. K. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.
25. N. L. Kleinman, J. C. Spall, and D. Q. Naiman. Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, 45(11):1570–1578, 1999.
26. H. J. Kushner and J. Yang. Stochastic approximation with averaging of the iterates: optimal asymptotic rates of convergence for general processes. *SIAM Journal on Control and Optimization*, 31:1045–1062, 1993.
27. H. J. Kushner and J. Yang. Stochastic approximation with averaging and feedback: rapidly convergent "on-line" algorithms. *IEEE Transactions on Automatic Control*, 40:24–34, 1995.
28. H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, New York, NY, 2003.
29. J. L. Maryak. Some guidelines for using iterate averaging in stochastic approximation. *Proceedings of the 36th IEEE Conference on Decision and Control*, 3:2287–2290, 1997.
30. A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.
31. B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.
32. H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.
33. D. Ruppert. A Newton-Raphson version of the multivariate Robbins-Monro procedure. *The Annals of Statistics*, 13(1):236–245, 1985.
34. D. Ruppert. Efficient estimations from a slowly convergent Robbins-Monro process. Technical report, Cornell University Operations Research and Industrial Engineering, Ithaca, NY, February 1988.
35. J. Sacks. Asymptotic distribution of stochastic approximation procedures. *The Annals of Mathematical Statistics*, 29(2):351–634, 1958.
36. P. Sadegh. Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation. *Automatica*, 33(5):889–892, 1997.
37. G. Sardis. Learning applied to successive approximation algorithms. *IEEE Transactions on Systems, Science and Cynbernetics*, SSC(6):97–103, 1970.
38. J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.
39. J. C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10):1839–1853, 2000.
40. J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley, Hoboken, NJ, 2003.

41. A. B. Tsybakov and B. T. Polyak. Optimal order of accuracy of search algorithms in stochastic optimization. *Problemy Peredachi Informatsii*, 26(2):45–63, 1990.
42. J. H. Venter. An extension of the Robbins-Monro procedure. *The Annals of Mathematical Statistics*, 38(1):181–190, 1967.
43. I. J. Wang and E. K. P. Chong. A deterministic analysis of stochastic approximation with randomized differences. *IEEE Transactions on Automatic Control*, 43(12):1745–1749, 1998.
44. Q. Wang. *Optimization with Discrete Simultaneous Perturbation Stochastic Approximation Using Noisy Loss Function Measurements*. PhD thesis, Johns Hopkins University, 2013.
45. S. Yakowitz, P. L'Ecuyer, and F. Vazquez-Abad. Global stochastic optimization with low-dispersion point sets. *Operations Research*, 48(6):939–950, 2000.
46. G. Yin and Y. M. Zhu. Almost sure convergence of stochastic approximation algorithms with nonadditive noise. *International Journal of Control*, 49(4):1361–1376, 1989.