Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

AN ADAPTIVE MULTIDIMENSIONAL VERSION OF THE KIEFER-WOLFOWITZ STOCHASTIC APPROXIMATION ALGORITHM

Mark Broadie
Deniz M. Cicek
Assaf Zeevi

Graduate School of Business
Columbia University
New York, NY 10027, USA

ABSTRACT

We extend the scaled-and-shifted Kiefer-Wolfowitz (SSKW) algorithm developed by Broadie, Cicek, and Zeevi (2009) to multiple dimensions. The salient feature of this algorithm is that it makes adjustments of the tuning parameters that adapt to the underlying problem characteristics. We compare the performance of this algorithm to the traditional Kiefer-Wolfowitz (KW) one and observe significant improvement in the finite-time performance on some stylized test functions and a multidimensional newsvendor problem.

1 INTRODUCTION

Simulation-optimization refers to optimization of functions whose values cannot be computed analytically and are evaluated via simulation. Because of the nature of simulation, the optimization uses noisy observations of the objective function. Stochastic approximation is one of the most widely studied simulation-optimization methods, since the pioneering papers by Robbins and Monro (1951) and Kiefer and Wolfowitz (1952) introduced the method. (See Kushner and Yin (2003) and references therein for a literature review.) Consider the following problem:

$$\max_{x\in\mathbb{R}^d} f(x) = \mathbb{E}[\widetilde{f}(x)]$$

where $\widetilde{f}$ is a noisy observation of $f:\mathbb{R}^d\to\mathbb{R}$. In order to solve this problem when $d=1$, Kiefer and Wolfowitz (1952) proposed the following stochastic approximation scheme, henceforth referred to as the KW algorithm:

$$X_{n+1} = X_n + a_n\left(\frac{\widetilde{f}(X_n+c_n)-\widetilde{f}(X_n-c_n)}{c_n}\right), \quad n=1,2,\ldots \tag{1}$$

where the "tuning sequences" $\{a_n\}$ and $\{c_n\}$ are real-valued and deterministic. Under the assumption that the function $f(x)$ has a unique point of maximum and is strongly concave, and further assumptions on the tuning sequences, Kiefer and Wolfowitz (1952) prove that the $\{X_n\}$ sequence converges in probability to $x^* = \arg\max_{x\in\mathbb{R}^d} f(x)$, the point of maximum. Because it may be too restrictive to impose such assumptions on the function $f(\cdot)$ over the entire domain, Kiefer and Wolfowitz (1952) argue that it is enough to have these functional assumptions hold on a compact set $I_0\subset\mathbb{R}^d$, which is known to include x^*, and they prescribe a truncation method that restricts function evaluations to this compact set (see Broadie, Cicek, and Zeevi (2009) for further discussion).

A multidimensional version of the KW algorithm was introduced by Blum (1954). His algorithm uses a one-sided finite-difference approximation of the gradient in each direction and can be described by the following recursion:

$$X^{(n+1)} = X^{(n)} + a^{(n)} \frac{\widetilde{g}(X^{(n)})}{c^{(n)}}, \quad n = 1,2,\ldots \tag{2}$$

Here $\widetilde{g}(X^{(n)}) = (\widetilde{f}(X^{(n)} + c^{(n)}e_1) - \widetilde{f}(X^{(n)}), \ldots, \widetilde{f}(X^{(n)} + c^{(n)}e_d) - \widetilde{f}(X^{(n)}))$, where $\{e_1, \ldots, e_d\}$ is the standard basis in $\mathbb{R}^d$. Blum (1954) proves that $X^{(n)}$, defined by the recursion in (2), converges almost surely to the point of maximum x^*. A version of this algorithm, where iterates are truncated in a similar manner to the original KW algorithm, will serve as the benchmark stochastic approximation procedure for comparison purposes. We call this version of the KW algorithm the *truncated Kiefer-Wolfowitz-Blum algorithm* (TKWB for short).

Stochastic approximation algorithms have been mostly studied from the perspective of their asymptotic performance. To that end, the choice $a^{(n)} = \alpha/(n+\beta)$ and $c^{(n)} = \gamma/n^{1/4}$, for some $\alpha, \gamma \in \mathbb{R}$ and $\beta \in \mathbb{Z}$, is seen to be optimal in the sense that it optimizes the rate of convergence of the mean-squared error (MSE), $\mathbb{E}[X^{(n)} - x^*]^2$, of the algorithm; see Dupac (1957) and Broadie, Cicek, and Zeevi (2009) for two different proofs. At the same time, the behavior of KW-type algorithms can be quite sensitive to the specification of these tuning sequences through the parameters α, β and γ; see Kim (2006) (page 163, the last paragraph of Section 4). To the best of our knowledge, there is no clear guidance in the literature on how to choose these parameters. A further issue is that if the tuning sequences do not "match" the characteristics of the underlying function and noise level, the theoretical convergence rates may no longer hold; see, e.g., Juditsky et al. (2007).

In this paper, we propose a multidimensional version of a KW-type algorithm, recently introduced by Broadie et al. (2009), that adaptively adjusts the tuning sequences to better match the problem structure. The algorithm uses suitable scaling and shifting of the tuning sequences, and will therefore be referred to as the *scaled-and-shifted* KW algorithm or SSKW for short. The benefit of using different tuning sequences along each dimension is explained. We illustrate numerically that the SSKW algorithm outperforms the TKWB algorithm for four test functions, each representative of a fundamental performance issue. We also compare the finite-time performance of the TKWB and SSKW algorithms on a well-studied simulation-optimization problem, the multidimensional newsvendor problem of Kim (2006) (Section 6, page 165).

The rest of the paper is organized as follows. Section 2 contains a brief discussion of the three possible issues that may result in poor finite-time behavior of the TKWB algorithm, and explains the remedies implemented in the SSKW algorithm. (The full description of the multidimensional SSKW algorithm is given in Appendix.) Section 3 describes the multidimensional newsvendor problem and includes finite-time performance comparisons of the TKWB and SSKW algorithms. We summarize the results and give concluding remarks in Section 4.

2 THE MULTIDIMENSIONAL SCALED AND SHIFTED KW ALGORITHM

Unlike the TKWB recursion given in (2), we use different $\{a_k^{(n)}\}$ and $\{c_k^{(n)}\}$ sequences in each dimension $k = 1, \ldots, d$. (For brevity, we will omit the subscript "k" when no confusion arises.) The motivation for this modification will be presented in Section 2.4. Using the same notation as in (2), the recursion for the SSKW algorithm is:

$$X_k^{(n+1)} = X_k^{(n)} + a_k^{(n)} \frac{\widetilde{f}(X^{(n)} + c_k^{(n)} e_k) - \widetilde{f}(X^{(n)})}{c_k^{(n)}}, \quad \text{for all } k = 1, \ldots, d \text{ and } n = 1,2,\ldots \tag{3}$$

KW-type algorithms are prone to poor finite-time performance. The main issues are as follows:

1. a long oscillatory period where the iterates oscillate between different boundaries of the truncation interval due to an $\{a^{(n)}\}$ sequence which is "too large,"
2. a degraded convergence rate of MSE due to setting the $\{a^{(n)}\}$ sequence "too small," and
3. iterates not exhibiting convergent behavior due to poor gradient estimates caused by a $\{c^{(n)}\}$ sequence which is "too small."

Next we illustrate these issues and explain how the SSKW algorithm addresses them. Put $a^{(n)} = \alpha/(n+\beta)$ and $c^{(n)} = \gamma/n$, for some constants $\alpha, \gamma \in \mathbb{R}$ and $\beta \in \mathbb{Z}$. The key is to adapt these three parameters without relying on knowledge of the gradient or the underlying function; the only information needed is a multidimensional interval $I_0 = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_d, u_d]$, which is known to contain the optimal solution x^*. We consider three different test functions in two dimensions. In all numerical illustrations, we use $I_0 = [-50, 50]^2$, $X_1 = (30, 30)$ and initially set $a_k^{(n)} = 1/n$, $c_k^{(n)} = 1/n^{1/4}$ for all $k = 1, \ldots, d$. These sequences will be used throughout the TKWB algorithm, and will be adaptively modified in the SSKW algorithm.

In each of the examples, the reported MSE results, slopes and medians are computed using 50,000 replications. Standard errors of estimates are calculated by using 50 independent batches of 1000 replications in each.

2.1 The Problem of a Long Oscillatory Period

A long oscillatory period is typically observed if the step-size sequence, $\{a^{(n)}\}$, is "large" relative to the gradient. This initial oscillatory behavior does not affect the asymptotic rate of convergence, but the algorithm may have poor finite-time performance, essentially until the $\{a^{(n)}\}$ sequence becomes suitably small. Figure 1(a) illustrates a single path of the first coordinate of the iterates in TKWB algorithm for the function $\widetilde{f}(x_1,x_2) = -(x_1^4 + x_2^4) + \varepsilon$ where ε is a standard normal random variable. In all numerical illustrations the noise ε is independent at each function evaluation. The median oscillatory period length in our tests was 4988 iterations.

Our remedy for this problem is to adaptively *shift* the $\{a^{(n)}\}$ sequence when any coordinate of the iterate $X^{(n)}$ falls outside its current truncation interval, i.e., if $X_k^{(n)} \notin I_k^{(n)} := [l_k, u_k - c_k^{(n)}]$ for any $k = 1,\ldots,d$, we calculate the integer β so that substituting $a_k^{(n)} := a_k^{(n+\beta)}$ in the iteration would keep the iterate within the truncation interval (the truncation interval uses l_k because one-sided finite difference approximation is used in equation (3)). Figure 1(c) plots a single path of the first coordinate of the iterates in SSKW algorithm, using the same function and same random numbers as in panel (a) of Figure 1. Using $a_1^{(n)} = 1/(n+4925)$ results in a significant decrease in the oscillatory period length to 34. (The median shift

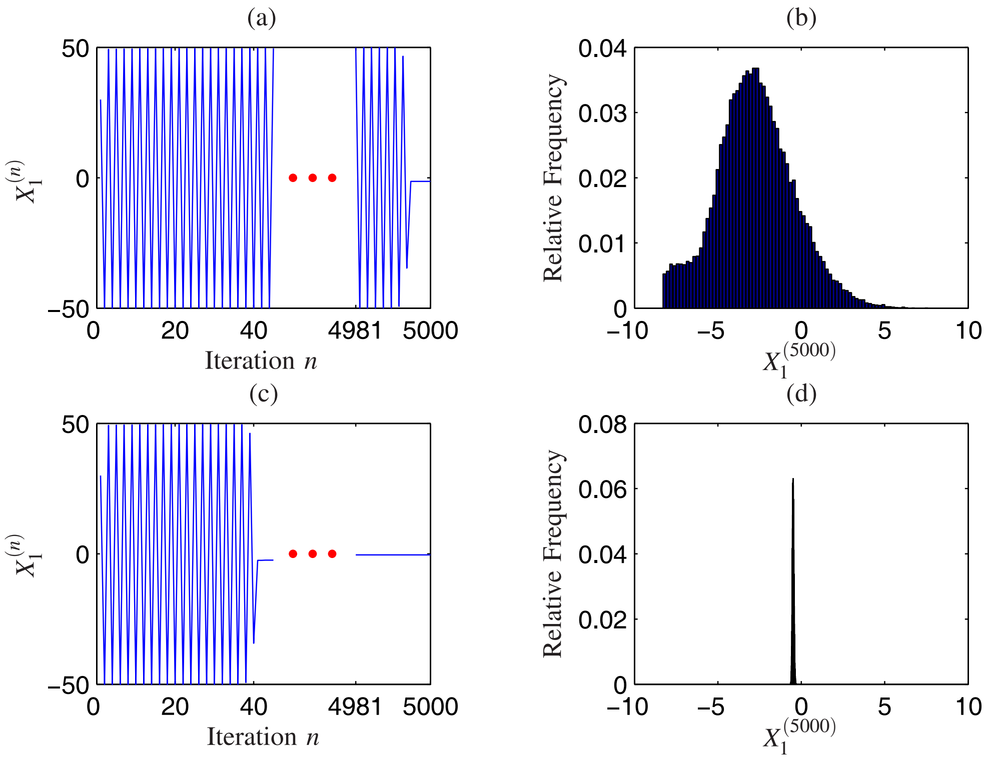

Figure 1: Oscillatory behavior. Panel (a) shows a sample path of the the first coordinate of iterates in the TKWB algorithm for the function $f(x_1,x_2) = -(x_1^4 + x_2^4)$ with $a^{(n)} = 1/n$, $c^{(n)} = 1/n^{1/4}$. The oscillatory behavior is observed for 4988 iterations. Panel (c) show a sample path in the SSKW algorithm in the same setting, using the same noise random sequence. The shift of $\beta = 4925$ corresponding to $a_1^{(n)} = 1/(n+4925)$ is finalized after 34 iterations. The sequence $c_1^{(n)} = 1/n^{1/4}$ is not scaled nor shifted. Panels (b) and (d) contain the relative frequency of $X_1^{(5000)}$ using 50,000 simulation replications for the TKWB and SSKW algorithms, respectively. Similar behavior is observed in the second coordinate.

and oscillatory period length were 4925 and 34, respectively.) Figures 1(b) and 1(d) show the histograms of $X_1^{(5000)}$, the first coordinate of the estimate of the point of maximum at iteration 5,000. Similar results hold for the second coordinate, and are omitted for brevity. The MSE at iteration 5000 for the TKWB algorithm is 26.11, whereas the corresponding value for the SSKW algorithm is 0.48 as a result of the decrease in the oscillatory period length.

2.2 The Problem of Degraded Convergence Rate

The tuning sequences $a^{(n)} = \alpha/(n+\beta)$ and $c^{(n)} = \gamma/n$ for some $\alpha, \gamma \in \mathbb{R}$ and $\beta \in \mathbb{Z}$, result in an optimal asymptotic MSE performance, $\mathbb{E}\|X^{(n)} - x^*\|^2 \sim 1/\sqrt{n}$, but this is predicated on α being set to be greater than a well-defined threshold that depends on the gradient of the unknown function. (See Juditsky et al. (2007) and Broadie et al. (2009) for formal statement of the condition and illustrative example.) Figure 2(a) shows the first coordinate of the iterates in TKWB algorithm for the function $\widetilde{f}(x_1,x_2) = -0.001(x_1^2+x_2^2)+\varepsilon$ where ε is normally distributed with zero mean and standard deviation of 0.001. Using a least squares fit of $\log(MSE)$ vs. $\log(n)$, from $n = 5{,}000$ to $n = 10{,}000$, the empirical convergence rate of the MSE is seen to be -0.004 (with a standard error of 5.6×10^{-8}).

In order to prevent this problem, we *scale up* the $\{a_k^{(n)}\}$ sequence adaptively over initial iterations of the algorithm, by increasing the constant α so that iterate n hits the boundary of the current truncation interval, $I_k^{(n)} = [l, u - c_k^{(n)}]$ in each dimension, i.e., either $X_k^{(n)} = l_k$ or $X_k^{(n)} = u_k - c_k^{(n)}$ for $k = 1,\dots,d$. Hence, during this initial phase of the algorithm, we force the iterates to oscillate between boundary points of the corresponding truncation interval in all dimensions. If at any

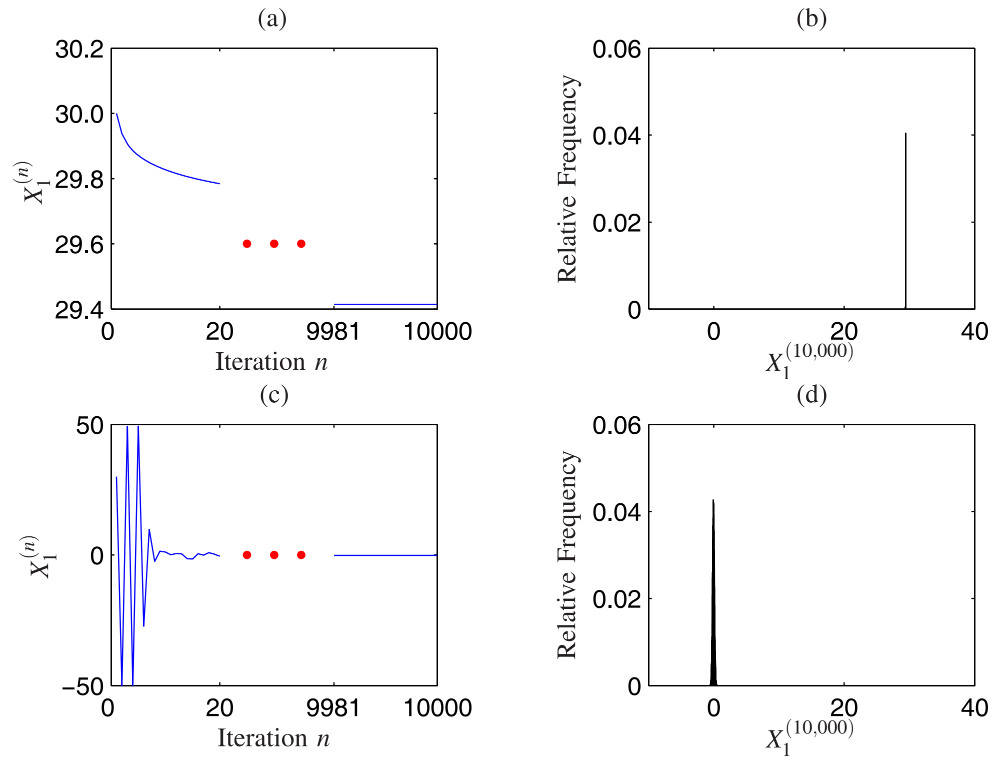

Figure 2: Degraded MSE convergence rate. Panel (a) shows a sample path of the the first coordinate of iterates in the TKWB algorithm for the function $f(x_1,x_2) = -0.001(x_1^2+x_2^2)$ with $a^{(n)} = 1/n$, $c^{(n)} = 1/n^{1/4}$. The convergence rate of the MSE is -0.004 (standard error is 5.6×10^{-8}). Panel (c) shows a sample path in the SSKW algorithm in the same setting, using the same noise random sequence. The adaptive scale up of $\alpha = 4022.2$ leading to $a_1^{(n)} = 4022.2/n$, is achieved after four iterations. Panels (b) and (d) show the relative frequency of $X_1^{(10,000)}$ using 50,000 simulation replications for the TKWB and SSKW algorithms, respectively.

iteration the gradient estimates are pointing towards the interior in some dimensions, say $D_1 \subseteq \{1,2,\ldots d\}$, and pointing outward in the other dimensions, we scale up $\{a_k^{(n)}\}$ for $k \in D_1$, iterate along these dimensions and re-estimate the gradient until the iterate oscillates from one boundary point to the opposite one in all dimensions.

Achieving this type of oscillation may require more than one estimate of the gradient along some dimensions. Suppose, for instance, we have a two-dimensional problem and at iteration n, $X^{(n)} = (l_1, u_2 - c_2^{(n)})$ and also suppose that the gradient estimate in both dimensions are positive. Because the iterate is at the lower boundary of the truncation interval along the first dimension and the gradient estimate is positive along this dimension, we can scale up $\{a_1^{(n)}\}$, if necessary, so that $X_1^{(n+1)} = u_1 - c_1^{(n+1)}$. So, we achieve oscillation along the first dimension. On the other hand, since the gradient is positive along the second dimension, it is "pushing" the iterate outside of the truncation interval. So we do not scale up $\{a_2^{(n)}\}$ at this point. Next, we iterate along the first dimension to the point $(u_1 - c_1^{(n+1)}, u_2 - c_2^{(n)})$ and re-estimate the gradient. Now suppose we have a gradient estimate which has a negative first component and positive second component. Because the gradient is again pushing the iterate outside of the truncation interval along the second dimension, we still do not scale-up $\{a_2^{(n)}\}$. So, oscillation in the second dimension is still not achieved. We again iterate along the first dimension (this time we do not change $\{a_1^{(n)}\}$ since it is already scaled up), say to point $(y, u_2 - c_2^{(n)})$ where $l_1 \le y \le u_1 - c_1^{(n+1)}$, and $l_1 \le y$ is satisfied by truncation, if necessary. We then estimate the gradient at $(y, u_2 - c_2^{(n)})$, iterate along the first dimension and if the second component of the gradient is now negative, then, to achieve the oscillation, we scale up $\{a_2^{(n)}\}$ so that $X_2^{(n+1)} = l_2$ and set $X^{(n+1)} = (y', l_2)$. With this, oscillation is achieved in all dimensions. (See Appendix A for further details.) Figure 2(c) illustrates a single path of the first coordinate of the iterates in SSKW algorithm, using the same function and same random numbers as in panel (a) of the same figure. In the SSKW algorithm, at the end of four "forced" boundary hits, the $\{a_1^{(n)}\}$ sequence becomes $4002.2/n$ (the median scale up factor is 4002). The rate of convergence is -0.501 (standard error is 0.009). Figures 2(b) and 2(d) give the histograms of $X_1^{(10{,}000)}$. We observe that final estimates of the SSKW algorithm have an MSE value of 0.047 at iteration 10,000 (standard error of estimate is 2×10^{-4}). The TKWB estimates are still close to the initial starting point of $X_1^1 = 30$, due to the step size being too small with respect to the gradient values. The MSE at iteration 10,000 for the TKWB algorithm is 1730.4 (standard error of estimate is 9×10^{-4}).

2.3 The Problem of Noisy Gradient Estimates

Excessively noisy function evaluations cause poor gradient estimates and result in iterates that do not exhibit convergent behavior for long periods. If the $\{c^{(n)}\}$ sequence is too small and the noise level is high, the difference in the function values in finite-difference estimates might be dominated by noise, and the iterates may move in random directions governed by the noise. Figure 3(a) illustrates the first coordinate of the iterates in TKWB algorithm for the function $\widetilde{f}(x_1,x_2) = 1000(\cos(\pi x_1/100) + \cos(\pi x_2/100)) + \varepsilon$ where ε is normally distributed with zero mean and standard deviation of 100. The MSE values at iterations 50, 500, 5000 and 10,000 are 1603, 992, 428 and 305, respectively.

Our remedy for this problem is to adaptively *scale up* the $\{c^{(n)}\}$ sequence to provide better gradient estimates. There are two sources of error in the finite difference estimation of the gradient. The *bias* which decreases as $\{c^{(n)}\}$ gets smaller, and the noise in the function evaluations for which a larger $\{c^{(n)}\}$ decreases the error. If the main source of the error in the gradient estimation is the function evaluation noise, scaling up might help by balancing the two sources of error. One of the possible indicators of noisy function evaluations causing poor gradient estimates is the existence of gradient estimates at the boundary which are pointing outside of the current truncation interval. Assuming the initial interval, I_0, contains the true point of optimum, the gradient at the boundaries of I_0 must point towards the interior of the truncation interval along some dimensions. If at some iteration n, the iterate is at the current truncation interval boundary along dimension k, and the gradient estimate in the next iteration is pointing outside of the truncation interval along the same dimension, then we scale up the $\{c_k^{(n)}\}$. For instance, if $X_k^{(n)} = l_k$, i.e., the iterate is at the lower boundary point, and the gradient estimate in the next iteration is negative, i.e., $\widetilde{f}(X^{(n)} + c_k^{(n)}) - \widetilde{f}(X^{(n)}) < 0$, then we set $c_k^{(n)} := \gamma_0 c_k^{(n)}$ and use this sequence for future iterations. The amount of scale up at every such instance, γ_0, is a user-defined parameter (the default value that we use in our numerical experiments is $\gamma_0 = 2$). Figure 3(c) illustrates a single path of the first coordinate of the iterates in SSKW algorithm, using the same function and same random numbers as in panel (a) of the same figure. At the end of iteration 9, the $\{a_1^{(n)}\}$ sequence is set to $19.39/n$ (the median scale up factor for this sequence is 10) and the $\{c_1^{(n)}\}$ sequence is set to $8/n^{1/4}$ (the median scale up factor for this sequence is also 8). The MSE at iterations 50, 500, 5000 and 10,000 are 696, 143, 44 and 31, respectively. The histograms of $X_1^{(10{,}000)}$ are shown in panels (b) and (d) of Figure 3.

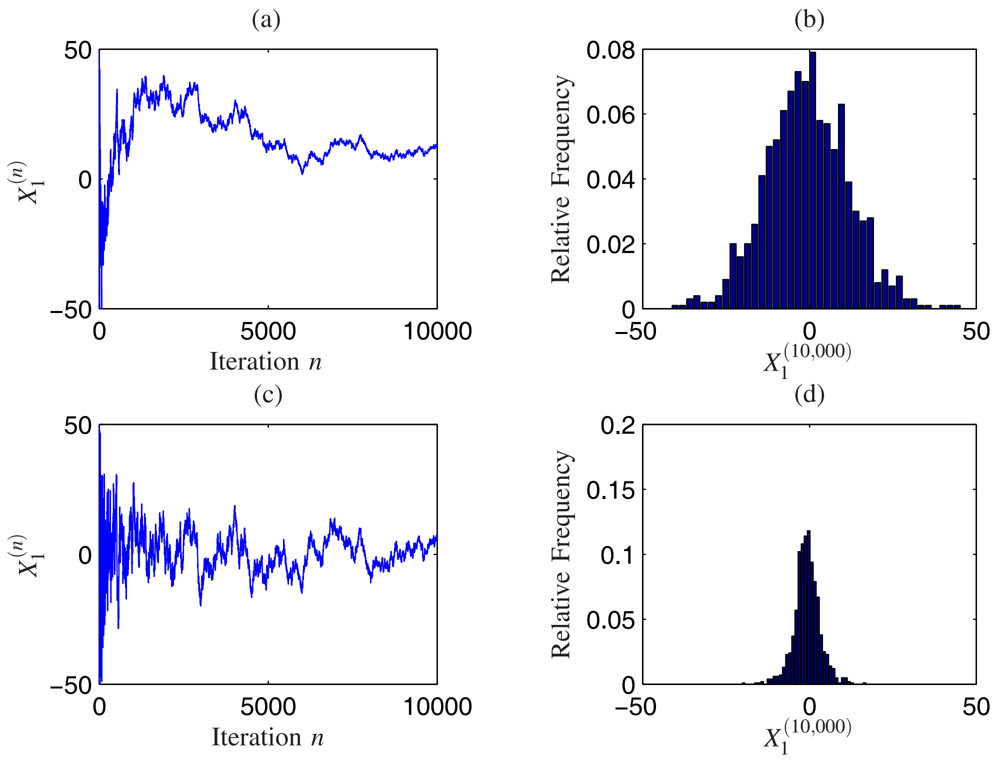

Figure 3: Noisy gradient estimates. Panel (a) shows a sample path of the the first coordinate of iterates in the TKWB algorithm for the function $f(x_1,x_2) = 1000(\cos(\pi x_1/100) + \cos(\pi x_2/100))$ with $a^{(n)} = 1/n$, $c^{(n)} = 1/n^{1/4}$. Panel (c) show a sample path in the SSKW algorithm in the same setting, using the same noise random sequence. At the end of iteration 9, the tuning sequences are adaptively changed to $a_1^{(n)} = 19.39/n$ and $c_1^{(n)} = 8/n^{1/4}$. Panels (b) and (d) contain the relative frequency of $X_1^{(10,000)}$ using 50,000 simulation replications for the TKWB and SSKW algorithms, respectively.

2.4 Different Tuning Sequences in Each Dimension

In general, we need different $\{a^{(n)}\}$ and $\{c^{(n)}\}$ sequences in each dimension to be able to adapt to different characteristics of the unknown function. For instance, if a two-dimensional function is "steep" along one dimension and "flat" in the other one, then, loosely speaking, in order to achieve good finite-time behavior we need a small $\{a^{(n)}\}$ sequence in the steep dimension and a large sequence in the flat one. Moreover, if we use the same $\{c^{(n)}\}$ sequence in both dimensions, a small $\{c^{(n)}\}$ sequence would result in gradient estimates being governed by the noise along the "flat" dimension since the true finite difference estimate of the gradient would be small. On the other hand, a large $\{c^{(n)}\}$ sequence would cause poor finite-difference estimates especially along the "steep" direction where we could have used a smaller $\{c^{(n)}\}$ sequence and get better gradient estimates.

In order to illustrate the disadvantage of using the same tuning sequences along all dimensions, we run the SSKW algorithm and a modified version of it (called "SSKW-1") that uses the same tuning sequences in all dimensions for the function $\widetilde{f}(x_1,x_2) = -0.001x_1^2 - x_2^4 + \varepsilon$ where ε is a standard normal random variable. Figure 4(a) and 4(b) illustrates the two coordinates of the iterates in a sample path of SSKW-1 algorithm. In both dimensions, with the forced boundary hits, iterates initially exhibit oscillatory behavior. Then, because of the steep gradient in the second dimension, the algorithm starts shifting the $\{a^{(n)}\}$ sequence to prevent oscillation, but this causes rate degradation along the first dimension where the function is too flat. After a few shifts, the change in the first coordinate at each iteration becomes too small, which results in almost constant iterate values in panel (a) of Figure 4. Figure 4(c) and 4(d) shows the two coordinates for the SSKW algorithm with different sequences. After all the adaptations, the sequences become $a_1^{(n)} = 2746/n$, $a_2^{(n)} = 1/(n+4884)$,

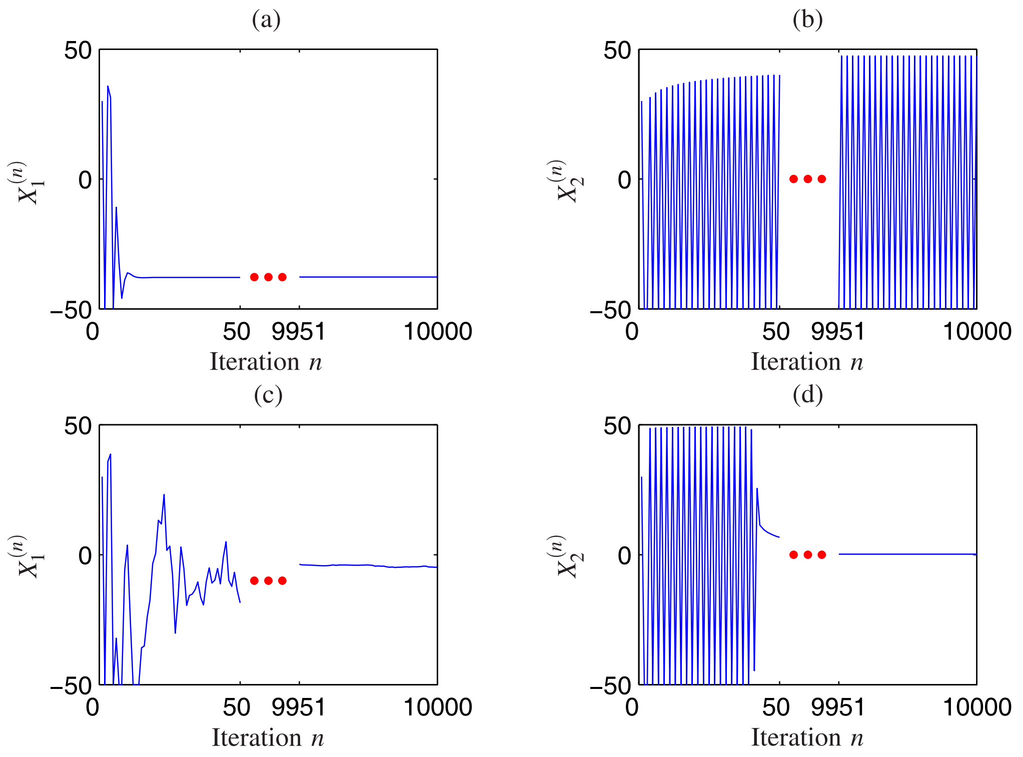

Figure 4: The effect of using the same tuning sequence in all dimensions. Panels (a) and (b) show a sample path of both coordinates of the iterates in the SSKW-1 algorithm using the same $\{a^{(n)}\}$ and $\{c^{(n)}\}$ sequences along all dimensions for the function $f(x_1,x_2) = -0.001x_1^2 - x_2^4$. Panels (c) and (d) show a sample path of the SSKW algorithm with different tuning sequences along all dimensions.

$c_1^{(n)} = 37.2/n^{1/4}$ and $c_2^{(n)} = 2/n^{1/4}$. Due to the high noise level relative to the gradient along the first dimension, the scale-up in the $\{c_1^{(n)}\}$ (the median scale-up is 32) is much larger than in $\{c_2^{(n)}\}$ (the median scale-up is 1). Figure 5 shows the histograms of the final estimates, $X_1^{(10,000)}$ and $X_2^{(10,000)}$ for SSKW-1 in panels (a) and (b), and for SSKW in panels (c) and (d). The MSE values at iteration 10,000 are 384 (with standard error of 2) and 35 (with standard error of 0.31) for SSKW-1 and SSKW algorithms, respectively.

3 APPLICATION TO A MULTIDIMENSIONAL NEWSVENDOR PROBLEM

In this section, we illustrate the performance of the SSKW algorithm on a multidimensional newsvendor problem introduced by Kim (2006) (Section 2, page 159). A manufacturer produces q products using p different resources. The manager decides on a non-negative resource vector $X \in \mathbb{R}_+^p$ and then observes the random demand, $D \in \mathbb{R}_+^q$, for each one of the products. Once the demand is known, the manager solves the following optimization problem to determine the product mix to manufacture:

$$\begin{array}{llcll} \max_{y\in\mathbb{R}_+^q} & v^T y & & & \\ s.t. & Ay & \leq & X & \text{(capacity constraints)} \\ & y & \leq & D & \text{(demand constraints),} \end{array}$$

where $v \in \mathbb{R}_+^q$ is the vector of profit margins for each product and A is a $p \times q$ matrix whose (i,j) component specifies the amount of resource i required to produce one unit of product j. Assuming $c \in \mathbb{R}_+^p$ is the cost vector for resources, the objective function of the manager is to maximize the expected maximal profit, $\Pi(X) = \mathbb{E}[v^T y^*(X,D)] - c^T X$. For numerical

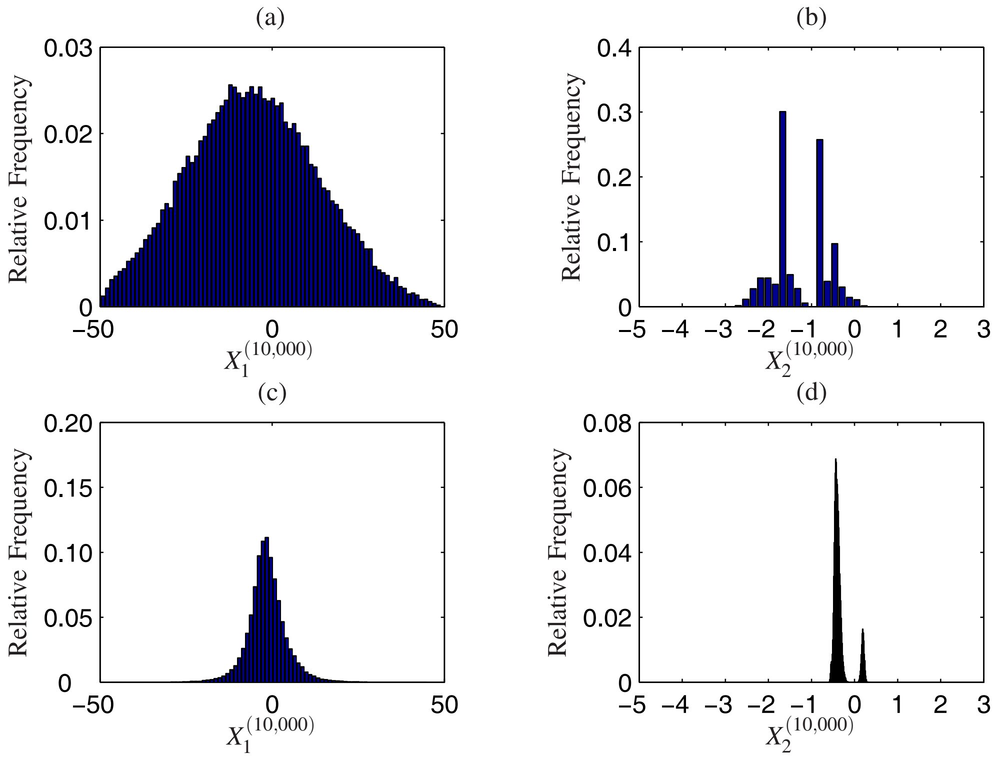

Figure 5: Using same tuning sequence in all dimensions. Panels (a) and (b) give the relative frequency of $X_1^{(10{,}000)}$ and $X_2^{(10{,}000)}$, respectively, using 50,000 simulation replications for the SSKW-1 algorithm for the function $f(x_1,x_2) = -0.001x_1^2 - x_2^4$. Panels (c) and (d) contain the corresponding histograms for the SSKW algorithm.

tests, we set $v = (6,5,4,3,2)$, c is a vector of ones and A is a matrix with all lower triangular entries equal to one and others zero. We assume the demand for the five products has a multivariate normal distribution with mean $(10,15,5,8,10)$, standard deviation $(5,10,2,3,6)$ and correlation matrix

$$\rho = \begin{pmatrix} 1 & -0.2 & 0.3 & 0.5 & 0.1 \\ -0.2 & 1 & -0.1 & -0.3 & -0.1 \\ 0.3 & -0.1 & 1 & 0.6 & 0.2 \\ 0.5 & -0.3 & 0.6 & 1 & 0.05 \\ 0.1 & -0.1 & 0.2 & 0.05 & 1 \end{pmatrix}.$$

In order to evaluate the function at any given resource level, we use 1000 independent draws of the demand and estimate the expected maximal profit. The point of maximum is estimated to be $x^* = (15,30,34,41,51)$ via an exhaustive grid search over integers using long simulation runs. The algorithms are set to run for 10,000 iterations and we use $25,000$ independent replications to estimate the MSE. The lower and upper bounds of the initial interval I_0 are set to be the resource levels required to meet the 30 and 99 percentile of the demand distribution, respectively. (The upper bound is (22, 61, 71, 86, 110) and the lower bound is (8, 18, 22, 29, 36).) Because no prior information about the objective function is assumed, the initial starting point of the algorithm, $X^{(1)}$, is set to be uniformly distributed between the upper and lower boundaries. The initial tuning sequences were set as $a^{(n)} = 1/n$ and $c^{(n)} = 1/n^{1/4}$. For this problem, we set the parameters $c^{(0)} = 0.1$ and $k_a = 10$ in the SSKW algorithm (see Appendix A).

Figure 6(a), (b) and (c) shows the estimation error in first, third and fifth coordinates in a sample path for TKWB algorithm. The MSE at iterations 50, 500, 5000 and 10,000 are 320, 216, 145 and 128, respectively. The main issue is the rate degradation due to using a too small $\{a^{(n)}\}$ sequence along third and fifth dimensions, which can be observed in panels (b) and (c) of

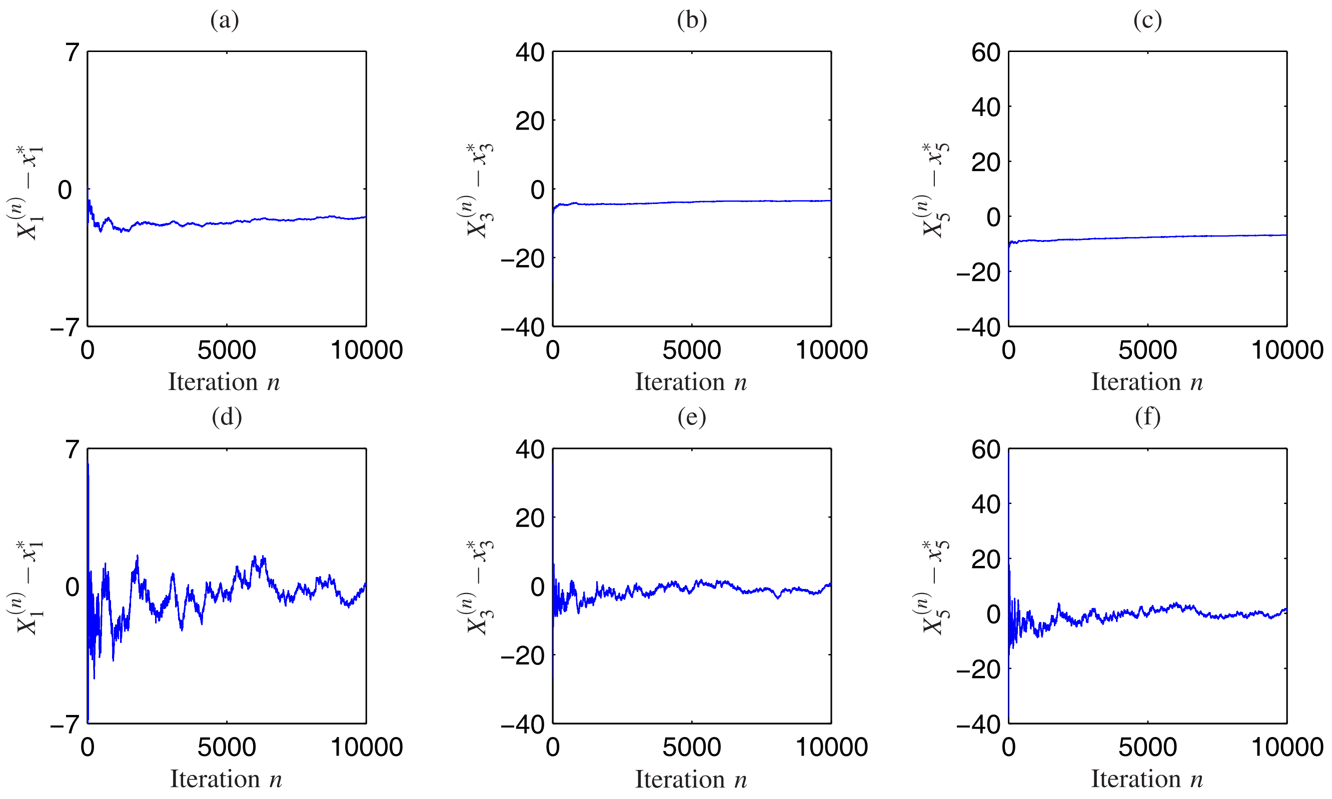

Figure 6: Multidimensional newsvendor problem. Panels (a), (b) and (c) show sample paths of the estimation error at each iteration in first, third and fifth coordinates in the TKWB algorithm, using $a^{(n)} = 1/n$ and $c^{(n)} = 1/n^{1/4}$. Panels (d), (e) and (f) show the corresponding sample paths in the SSKW algorithm using same random number sequence.

Figure 6. Panels (d), (e) and (f) in the same figure show the corresponding plots for the SSKW algorithm. At the end of iteration 52, the adaptations of the tuning sequences are finished and they are set as $a_1^{(n)} = 28.37/(n+10)$, $a_2^{(n)} = 252.21/(n+23)$, $a_3^{(n)} = 69.82/n$, $a_4^{(n)} = 577.97/(n+70)$, $a_5^{(n)} = 95.17/n$, $c_1^{(n)} = 3.48/n^{1/4}$, $c_2^{(n)} = 7.65/n^{1/4}$, $c_3^{(n)} = 4/n^{1/4}$, $c_4^{(n)} = 12.05/n^{1/4}$ and $c_5^{(n)} = 4/n^{1/4}$. The resulting MSE values at iterations 50, 500, 5000 and 10,000 are 535, 101, 36 and 27, respectively. Figure 7 shows the relative frequency of estimation error at iteration 10,000.

4 CONCLUDING REMARKS

We have tested our algorithm on five different test cases, four of which illustrate potentially problematic objective functions, while the fifth is a well-studied operations management problem. In all our tests, the proposed Scaled-and-Shifted KW (SSKW) algorithm outperforms the benchmark truncated Kiefer-Wolfowitz algorithm. The summary of the results are provided in Table 1. (The convergence rate estimate for TKWB algorithm in the first and fourth cases in the table are presented for completeness but are actually not valid estimates for asymptotic convergence rate due to the long oscillatory period.) The improvement in MSE values at iteration 5,000 range from a factor of 4 to over 25,000. The convergence rate estimates show improvement in most cases. The performance improvement is achieved by adaptive adjustments of the constants $\alpha, \gamma \in \mathbb{R}$ and $\beta \in \mathbb{Z}$ that are used in the tuning sequences $\{a^{(n)}\}$ and $\{c^{(n)}\}$. The median estimates for the tuning sequence constants are given in Table 2. The estimations use 25,000 replications of the algorithms.

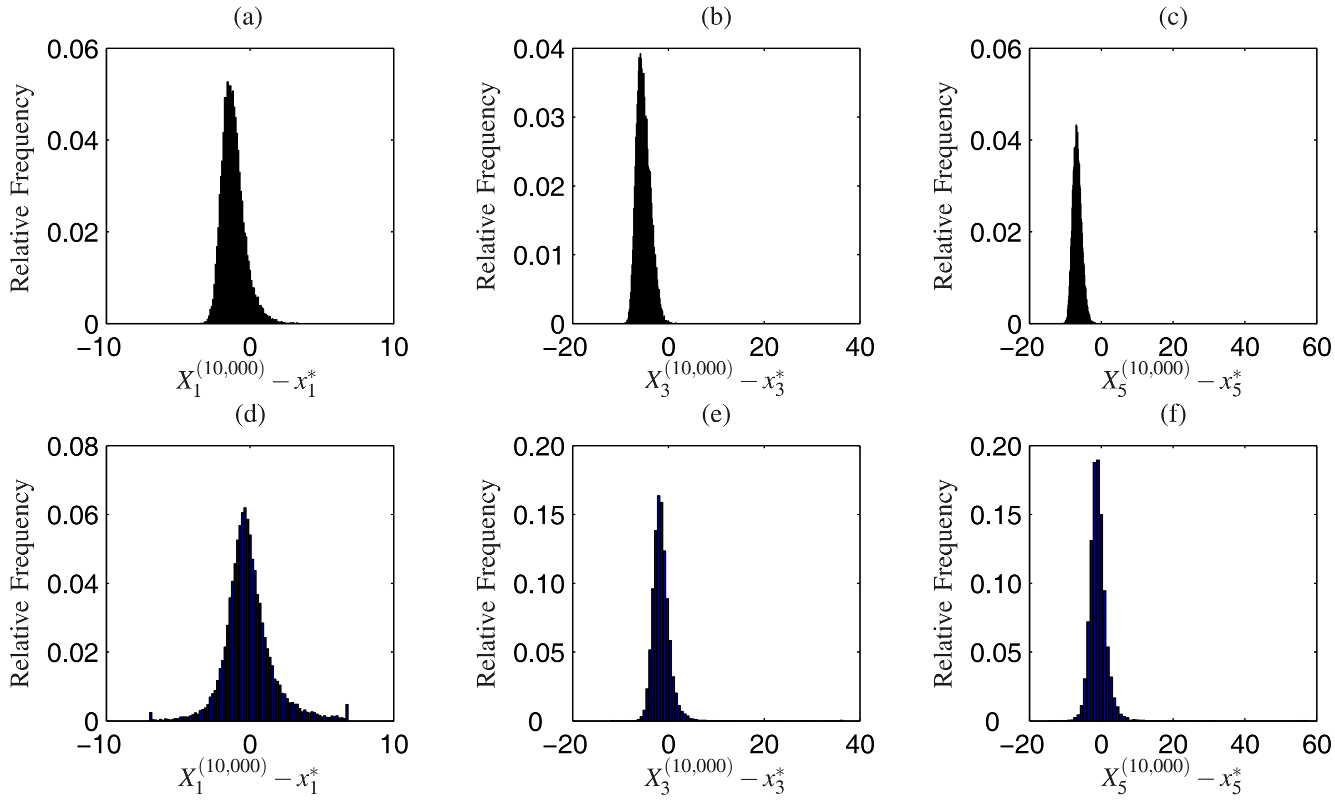

Figure 7: Multidimensional newsvendor problem. Panels (a)-(c) give the relative frequency of $X_1^{(10,000)} - x_1^*$, $X_3^{(10,000)} - x_3^*$ and $X_5^{(10,000)} - x_5^*$, using 25,000 simulation replications for the TKWB algorithm. Panels (d)-(f) show the corresponding histograms for the SSKW algorithm.

Table 1: Summary of results. This table provides the MSE values at iterations 50, 500 and 5000 and the convergence rate estimates for the five different test cases. The standard errors of MSE estimates are all within 2% of the MSE values and the standard errors of the rate estimates are provided in parentheses. The last column gives the median oscillatory period length along the first dimension for SSKW algorithm and the overall oscillatory period length in TKWB algorithm.

			MSE		Convergence	Median oscillatory
Function	Algorithm	50	500	5000	Rate	period length
$-x_1^4 - x_2^4$	SSKW	17.4	2.95	0.48	-0.62 (0.002)	34
	TKWB	5000	5000	26.11	-3.27 (0.002)	4988
$-0.001(x_1^2 + x_2^2)$	SSKW	0.72	0.21	0.066	-0.50 (0.009)	4
	TKWB	1767.7	1751.3	1735.2	-0.004 (5×10^{-8})	0
$1000(\cos(\pi x_1/100)+$	SSKW	696	143	44	-0.49 (0.01)	6
$\cos(\pi x_2/100))$	TKWB	1603	992	428	-0.49 (0.009)	10
$-0.001x_1^2 - x_2^4$	SSKW	461	143	48	-0.45 (0.01)	5
	SSKW-1	2615	2573	1170	-1.53 (0.01)	4236
Multidimensional	SSKW	535	101	36	-0.46 (0.07)	6
newsvendor	TKWB	320	216	145	-0.17 (0.004)	0

Table 2: Summary of adaptive changes to tuning sequences. The table on the left provides the median estimates for the constants α, β and γ in the tuning sequences along the first dimension in the first three test cases and in both dimensions in the last test case. The table on the right provides the median estimates for the constants in the tuning sequences in the multidimensional newsvendor problem.

Function	α	β	γ
$-x_1^4 - x_2^4$	1	4925	1
$-0.001(x_1^2 + x_2^2)$	4002	0	1
$1000(\cos(\pi x_1/100) + \cos(\pi x_2/100))$	10.0	0	8
$-0.001x_1^2 - x_2^4$ (dimension 1)	1493	0	32
$-0.001x_1^2 - x_2^4$ (dimension 2)	1	4925	1

Dimension	α	β	γ
1	41	186	2
2	118	9	6
3	163	14	8
4	239	30	8
5	286	14	8

A MULTIDIMENSIONAL SSKW ALGORITHM

Here we present the multidimensional SSKW algorithm. The MATLAB implementation can be downloaded from <www.columbia.edu/∼mnb2/broadie/research.html>.

Step 1. Set algorithm parameters

- h_0: the number of forced hits to boundary points l_k and $u - c_k^{(n)}$ for all $k = 1, \ldots, d$ by scaling up the $\{a^{(n)}\}$ sequence (sample default: 4).
- γ_0: the scaling up factor for the $\{c^{(n)}\}$ sequence (sample default: 2).
- k_a: an upper bound on the number of shifts in the $\{a^{(n)}\}$ sequence (sample default: 50).
- υ_a: an initial upper bound on the shift in the $\{a^{(n)}\}$ sequence (sample default: 10).
- k_c: an upper bound on the number of scale-ups in the $\{c^{(n)}\}$ sequence (sample default: 50).
- $c^{(0)}$: the parameter defining the maximum possible value of $\{c^{(n)}\}$ sequence after the scale-ups; i.e., $c_k^{(n)} \leq c_k^{max} = c^{(0)}(u_k - l_k)$ for all $k = 1, \ldots, d$, $n \geq 1$ (sample default: 0.2).
- ζ: the number of consecutive iterations of no $\{c^{(n)}\}$ scale-ups that we need to achieve in order to start shifting the $\{a^{(n)}\}$ sequence (sample default: 25).
- m^{max}: an upper bound on the iteration number of the last adaptation in the sequences, i.e., after iteration m^{max} no scaling nor shifting on the sequences is done; we require $m^{max} \geq h_0$ (sample default: total number of iterations).
- g_{max}: the maximum number of gradient estimations allowed to achieve oscillation along each dimension (sample default: 20).
- $X^{(1)}$: initial starting point; can be random or deterministic but must satisfy $X_k^{(1)} \in [l_k, u_k - c_k^{(1)}]$ for all $k = 1, \ldots, d$.

Step 2: For $n \leq h_0$,

(a) Calculate $X^{(n+1)}$ using the recursion given in (3).

(b) Scale up the $\{a_k^{(n)}\}$ sequence for each $k = 1, \ldots, d$, if necessary, ensuring a different truncation interval boundary point is hit in every dimension. Set $g = 1$ (counter for number of gradient estimations) and $S_{osc} = \{1, \ldots, d\}$ (the set of dimensions along which oscillation is not complete). While $g \leq g_{max}$ and $|S_{osc}| > 0$, set $\widetilde{X}^{(n+1)} = X^{(n+1)}$. Then, for each $k \in S_{osc}$

(i) If $X_k^{(n+1)} < u_k - c_k^{(n)}$ and $X_k^{(n+1)} > X_k^{(n)}$, find the scale-up factor for $\{a_k^{(n)}\}$ sequence that makes $X_k^{(n+1)} = u_k - c_k^{(n+1)}$; i.e., set $\alpha = (u_k - c_k^{(n+1)} - X_k^{(n)})/(X_k^{(n+1)} - X_k^{(n)})$ and then use the new sequence $\{a_k^{(n)}\} \leftarrow \{\alpha a_k^{(n)}\}$ for the rest of the iterations. Set $\widetilde{X}_k^{(n+1)} = u_k - c_k^{(n+1)}$, $S_{osc} = S_{osc} \setminus \{k\}$.

(ii) If $X_k^{(n+1)} > l_k$ and $X_k^{(n+1)} < X_k^{(n)}$, find the scale-up factor for $\{a_k^{(n)}\}$ sequence that makes $X_k^{(n+1)} = l_k$; i.e., set $\alpha = (l_k - X_k^{(n)})/(X_k^{(n+1)} - X_k^{(n)})$ and then use the new sequence $\{a_k^{(n)}\} \leftarrow \{\alpha a_k^{(n)}\}$ for the rest of the iterations. Set $\widetilde{X}_k^{(n+1)} = l_k$, $S_{osc} = S_{osc} \setminus \{k\}$.

(iii) Scale up the $\{c_k^{(n)}\}$ sequence whenever the gradient estimate is too noisy: If $X_k^{(n+1)} > X_k^{(n)} = u_k - c_k^{(n)}$ or $X_k^{(n+1)} < X_k^{(n)} = l_k$ then calculate $\gamma = \min\{\gamma_0, c^{max}/c_k^{(n)}\}$ and use the new sequence $\{c_k^{(n)}\} \leftarrow \{\gamma c_k^{(n)}\}$ for the rest of the iterations.

(iv) Re-calculate $X^{(n+1)}$ using the recursion given in (3) with $X^{(n)} = \widetilde{X}^{(n+1)}$. Update $g = g+1$.

(c) Set $X_k^{(n+1)} = \min\{u_k - c_k^{(n+1)}, \max\{\widetilde{X}_k^{(n+1)}, l_k\}\}$ for all $k = 1,\dots,d$. Increment n.

Step 3: For $n > h_0$,

(a) Calculate $X^{(n+1)}$ using the recursion given in (3).

(b) Shift the $\{a_k^{(n)}\}$ sequence for all $k = 1,\dots,d$, if necessary, to prevent iterates exiting the truncation interval, but use the upper bound parameter υ_a to prevent a too large shift.

(i) If $X_k^{(n+1)} > u_k - c_k^{(n+1)}$, $X_k^{(n)} < u_k - c_k^{(n)}$ then find the minimum shift in $\{a_k^{(n)}\}$ sequence that makes $X_k^{(n+1)} \leq u_k - c_k^{(n+1)}$; i.e.,

- Solve $\max(u_k - c_k^{(n+1)} - X_k^{(n)}, \upsilon_a)/\left(\frac{\widetilde{f}(X^{(n)}+c_k^{(n)}e_k) - \widetilde{f}(X^{(n)})}{c_k^{(n)}}\right) = a_k^{(n+\beta)}$ for β. If $\beta = \upsilon_a$, then set $\upsilon_a = 2\upsilon_a$.
- Use $\{a_k^{(n)}\} \leftarrow \{a_k^{(n+\lceil\beta\rceil)}\}$ for the rest of the iterations

(ii) If $X_k^{(n+1)} < l_k$, $X_k^{(n)} > l_k$ then find the minimum shift in $\{a_k^{(n)}\}$ sequence that makes $X_k^{(n+1)} \leq l_k$; i.e.,

- Solve $\max(l_k - X_k^{(n)}, -\upsilon_a)/\left(\frac{\widetilde{f}(X^{(n)}+c_k^{(n)}e_k) - \widetilde{f}(X^{(n)})}{c_k^{(n)}}\right) = a_k^{(n+\beta)}$ for β. If $\beta = \upsilon_a$, then set $\upsilon_a = 2\upsilon_a$.
- Use $\{a_k^{(n)}\} \leftarrow \{a_k^{(n+\lceil\beta\rceil)}\}$ for the rest of the iterations

(c) Repeat Step 2(b)(iii). Set $X_k^{(n+1)} = \min\{u_k - c_k^{(n+1)}, \max\{X_k^{(n+1)}, l_k\}\}$ for all $k = 1,\dots,d$. Increment n.

REFERENCES

Blum, J. 1954. Multidimensional stochastic approximation methods. *The Annals of Mathematical Statistics* 25 (4): 737–744.

Broadie, M., D. Cicek, and A. Zeevi. 2009. General bounds and finite-time improvements for stochastic approximation algorithms. Working paper. Columbia University.

Dupac, V. 1957. O Kiefer-Wolfowitzově approximační methodě. *Časopis pro Pěstování Matematiky* 82:47–75.

Juditsky, A., G. Lan, A. Nemirovski, and A. Shapiro. 2007. Stochastic approximation approach to stochastic programming. *Submitted to SIAM Journal on Optimization*.

Kiefer, J., and J. Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics* 23 (3): 462–466.

Kim, S. 2006. Gradient-based simulation optimization. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. Perrone, F. Wieland, J. Liu, B. Lawson, D. Nicol, and R. Fujimoto, 159–167. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kushner, H., and G. Yin. 2003. *Stochastic approximation and recursive algorithms and applications*. New York: Springer-Verlag.

Robbins, H., and S. Monro. 1951. A stochastic approximation method. *The Annals of Mathematical Statistics* 22 (3): 400–407.

AUTHOR BIOGRAPHIES

MARK BROADIE is the Carson Family Professor of Business in the Graduate School of Business, Columbia University. His research focuses on issues in financial engineering, with a particular focus on the design and analysis of efficient Monte Carlo methods for pricing and risk management. His email address is <mnb2@columbia.edu> and his webpage is www.columbia.edu/∼mnb2/broadie/.

DENIZ M. CICEK is a Ph.D. Candidate in the Decision, Risk and Operations division of Columbia Business School. His research interests are simulation optimization and its applications. His email address is <dcicek05@gsb.columbia.edu>.

ASSAF ZEEVI is the Kravis Professor of Business in the Graduate School of Business, Columbia University. His main research focuses on stochastic models of service systems, revenue management, simulation, statistics and applied probability. His email address is <assaf@gsb.columbia.edu>.