ABSTRACT

Title of dissertation: STOCHASTIC SIMULATION: NEW STOCHASTIC APPROXIMATION METHODS AND SENSITIVITY ANALYSES

Marie Chau, Doctor of Philosophy, 2015

Dissertation directed by: Prof. Michael C. Fu
R.H. Smith School of Business
& Institute of Systems Research

In this dissertation, we propose two new types of stochastic approximation (SA) methods and study the sensitivity of SA and of a stochastic gradient method to various input parameters. First, we summarize the most common stochastic gradient estimation techniques, both direct and indirect, as well as the two classical SA algorithms, Robbins-Monro (RM) and Kiefer-Wolfowitz (KW), followed by some well-known modifications to the step size, output, gradient, and projection operator.

Second, we introduce two new stochastic gradient methods in SA for univariate and multivariate stochastic optimization problems. Under a setting where both direct and indirect gradients are available, our new SA algorithms estimate the gradient using a hybrid estimator, which is a convex combination of a symmetric finite difference-type gradient estimate and an average of two associated direct gradient estimates. We derive variance minimizing weights that lead to desirable theoretical properties and prove convergence of the SA algorithms.

Next, we study the finite-time performance of the KW algorithm and its sen-

sitivity to the step size parameter, along with two of its adaptive variants, namely Kesten's rule and scale-and-shifted KW (SSKW). We conduct a sensitivity analysis of KW and explore the tightness of an mean-squared error (MSE) bound for quadratic functions, a relevant issue for determining how long to run an SA algorithm. Then, we propose two new adaptive step size sequences inspired by both Kesten's rule and SSKW, which address some of their weaknesses. Instead of using one step size sequence, our adaptive step size is based on two deterministic sequences, and the step size used in the current iteration depends on the perceived proximity of the current iterate to the optimum. In addition, we introduce a method to adaptively adjust the two deterministic sequences.

Lastly, we investigate the performance of a modified pathwise gradient estimation method that is applied to financial options with discontinuous payoffs, and in particular, used to estimate the Greeks, which measure the rate of change of (financial) derivative prices with respect to underlying market parameters and are central to financial risk management. The newly proposed kernel estimator relies on a smoothing bandwidth parameter. We explore the accuracy of the Greeks with varying bandwidths and investigate the sensitivity of a proposed iterative scheme that generates an estimate of the optimal bandwidth.

STOCHASTIC SIMULATION: NEW STOCHASTIC APPROXIMATION METHODS AND SENSITIVITY ANALYSES

by

Marie Chau

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Prof. Michael C. Fu, Chair/Advisor
Prof. Steven I. Marcus
Prof. Kasso A. Okoudjou
Prof. Ilya O. Ryzhov
Prof. Paul J. Smith

© Copyright by
Marie Chau
2015

To my loving and supportive parents.

Acknowledgments

First, I would like to express my sincerest gratitude and deepest appreciation to my advisor, Prof. Michael C. Fu, for the tremendous amount of support, guidance, feedback, and faith throughout this entire journey. Without him, this dissertation would not be possible. Prof. Fu is not only a highly-respected scholar and an excellent teacher who genuinely cares about his students, but also an amazing mentor, always with the best intentions. I admire his breadth and depth of knowledge, patience, humility, authenticity, honesty, compassion, optimism, and innate good-hearted nature. For this experience and the opportunity to work with him, I will be forever grateful.

Next, I would like to thank the other committee members - Professors Steve I. Marcus, Kasso A. Okoudjou, Ilya O. Ryzhov, and Paul J. Smith - for taking the time to read this dissertation and attend my defense. Special thanks to Professors Patrick M. Fitzpatrick, Eric V. Slud, and David H. Hamilton for their encouragement and willingness to help in their classes as well as in my job search.

Last, but certainly not least, I would like to thank the people I've met at UMD who have since become some of my nearest and dearest friends. A very special thanks to Patrick Sodré Carlos who I met randomly in differential equations during summer school in 2007 and is now one of my closest friends and biggest supporters. He's always willing to help not only myself but others as well, and is one of the main reasons why I'm still here. Another special thanks to Karamatou Yacoubou Djima for her loyal friendship from day one of boot camp. Graduate school would not

have been the same without her, from our countless food outings, endless hours of studying for quals, interesting conversations, fun times, and many laughs. Also a very special thanks to Huashuai Qu, Xuan Liu, Jong Jun Lee, Zhixin Lu, Ran Ji, and Anusha Dandapani for being amazing! Each of you have made a dent in my life in your own special way. Thanks to the rest of my 1305 officemates. Many thanks to Rhyneta Gumbs who has a very kind heart, always looking out for the students. Thanks to the other girl's night math ladies, Jennifer Clarkson, Clare Wickman, and Hana Ueda for your emotional support and regular dinner outings. Also, thanks to the rest of the boot camp class of 2008 for unforgettable memories. Thanks to Hisham Talukder for the many laughs, Temba and Joe for our fun outings, Lucia Simonelli for our salsa nights, Yimei Fan for fun times, Dana Botesteanu for good conversations over dinner/drinks, Changhui Tan and Wenqing Hu for their patience and invaluable math discussions. Also, thanks to Alverda McCoy for always being so pleasant and flexible, saving me from many administrative disasters.

Most of all, I thank my parents. Words can't begin to explain how grateful I am for all they've done throughout my entire life. Their morals, values, determination, patience, and supportive nature have shaped me into the person I am today.

Table of Contents

List of Figures

List of Notations

$\mathbb{R}$	the field of real numbers
d	number of dimensions
Θ	parameter space (continuous)
θ	continuous parameter
$f(\cdot)$	distribution of input parameters (excl. ch. 3, 4)
$J(\cdot)$	function of interest
c_n	perturbation size
$\mathbf{\Delta}_n$	random vector
$Y(\cdot, \cdot)$	sample performance measure (ch. 1, 2)
$\tilde{f}$	sample performance measure (ch. 3)
a_n	step or gain size
$\mathbf{e}_i$	ith unit basis vector
ξ	random effects
$\|\| \cdot \|\|$	Euclidean norm
$\langle \cdot, \cdot \rangle$	dot product

Chapter 1

Introduction

Consider the stochastic optimization problem

$$\min_{\mathbf{x}\in\Theta} J(\mathbf{x}), \tag{1.1}$$

where $J : \mathbb{R}^d \to \mathbb{R}$, $J(\mathbf{x}) = E[Y(\mathbf{x}, \xi)]$, and ξ denotes the stochastic effects. In simulation optimization, the objective is to find $\mathbf{x}^*$ that minimizes the function of interest $J(\mathbf{x})$. The search space Θ is, in general, either continuous or discrete, but we only focus on the *continuous* case, which restricts the range of methods that can be applied. One of the most useful and well-known methods in simulation optimization for continuous parameters is stochastic approximation (SA), which generates a sequence of estimates $\{\mathbf{x}_n\}$ converging to a solution of $\nabla J(\mathbf{x}) = \mathbf{0}$ using the recursion

$$\mathbf{x}_{n+1} = \Pi_\Theta \left(\mathbf{x}_n - a_n \widehat{\nabla} J(\mathbf{x}_n) \right), \tag{1.2}$$

where $\Pi_\Theta(\mathbf{x})$ is a projection of $\mathbf{x}$ back into the feasible region Θ if $\mathbf{x} \notin \Theta$, a_n is a positive step size or gain size, $\widehat{\nabla} J(\mathbf{x}_n)$ is an estimate of the true gradient $\nabla J(\mathbf{x}_n)$, and $\mathbf{x}_N$ is the output, where N is the stopping time, which we denote by $\mathbf{x}_N^*$.

The first SA algorithm introduced in [50] estimated the true gradient using an unbiased direct gradient, which unfortunately, is not always available in practice, so [37] proposed the use of indirect gradients (i.e., finite differences), which only

require sample performances. SA is simple and requires very little memory due to its recursive nature, and thus can be implemented as an online method. As a result, SA algorithms are used in a wide variety of application areas such as signal processing, statistics, operations research, and machine learning [9].

The landmark papers, [50] and [37], proved convergence in probability for the one-dimensional classical Robbins-Monro (RM) and Kiefer-Wolfowitz (KW) methods, respectively. Later, [5] modified the original conditions of the convergence theorems for both RM and KW to obtain almost sure (a.s.) convergence in higher dimensions. Then, [14] and [53] proved asymptotic normality, and [20] established asymptotic normality for general SA algorithms in the multidimensional setting. More recently, the focus has shifted to finite-time error bounds such as the mean-squared error (MSE) of the estimate $E[||\mathbf{x}_N^* - \mathbf{x}^*||^2]$, where $||\cdot||$ denotes the Euclidean norm [7, 65], or the difference between the objective value at the estimate and the optimal objective value [31, 45, 46].

Stochastic approximation converges asymptotically under certain conditions, but the practical performance depends highly on the components of recursion (1.2). Methods have been introduced to increase its robustness, such as choosing appropriate deterministic or adaptive step sizes [7, 21, 36, 51, 59, 65], generating estimates based on a subset of iterates [48, 49, 52], projecting iterates back into the feasible region in a clever/strategic manner [2], and stabilizing gradients or increasing gradient accuracy. By exploiting previous gradient estimates, [27], [47], and [63] propose gradient averaging methods to increase gradient stability. Moreover, [1] and [3] present methods to prevent gradients from taking extreme values, and [7] introduce

Figure 1.1: Illustration: Generating sample performance

a technique that reduces the variance by increasing the perturbation size of finite differences adaptively.

The SA algorithms with the fastest convergence rates are not necessarily superior. Biased gradient estimates lead to slower asymptotic convergence rates, but could outperform unbiased direct gradients in finite-time. One major factor in the performance is the choice in step size. It is difficult to select an appropriate step size since the function (1.1) is unknown, and it is impossible to find a universally optimal step size for all situations. Therefore, adaptive step sizes that adjust based on the ongoing performance of the algorithm have been proposed to tackle the issue.

In the simulation optimization context, the gradient estimate $\widehat{\nabla} J(\mathbf{x}_n)$ is assumed to be computationally expensive to generate. Each sample performance $Y(\mathbf{x}, \xi)$ is generated through a simulator, as illustrated in Figure 1.1, which can be used to estimate the gradient. In some cases, the simulator can also generate an estimate of the sample performance gradient $\nabla Y(\mathbf{x}, \xi)$ simultaneously, but it requires additional information about the system dynamics, which is not always available. Each simulation run is computationally heavy; therefore, the number of iterations N is limited, so the final parameter estimate does not have any guarantees based on the asymptotic theory. The expensive gradient together with the step size sensitivity reinforces the importance of finite-time performance.

The sensitivity of the function of interest to the parameters is also critical, and the category/class of methods for such gradient approximations is called sensitivity analysis. The main objective is to estimate the sensitivity of $J(\mathbf{x})$ accurately, so the key is to generate an accurate and reliable gradient estimate. Unfortunately, the function $J(\mathbf{x})$ could have undesirable properties such as discontinuities, which restricts the applicable methods. Common gradient estimates and their applicability will be discussed in Section 2.1.

In finance, the accuracy of the sensitivity estimates of hedging tools is critical. For example, an option is a financial instrument that can be used to hedge risk, and its sensitivity to market parameters is useful in making investment decisions. If the price of the underlying stock hits the strike price, then the option can be exercised, resulting in some positive payoff. However, if the stock does not reach the strike price, then it cannot be exercised, resulting in zero profit. The payoff function of an option may be discontinuous, which adds a level of difficulty in estimating the gradient with respect to market parameters, also known as Greeks. The well-known infinitesimal perturbation analysis (IPA) is not applicable in this case, but a modified version, smoothed perturbation analysis (SPA), can be applied to discontinuous functions [28]. More recently, kernel estimators have been used in combination with IPA estimators to generate a hybrid gradient or modified IPA [43]. The kernel relies on a bandwidth parameter chosen by the user, and [43] proposed a selection method based on minimizing the asymptotic MSE, which has several input parameters, again determined by the user.

The commonality between stochastic approximation (SA) and sensitivity anal-

ysis lies in their requirement for a gradient estimate, although serving different purposes. Many of the stochastic gradient estimation methods have associated asymptotic results, which are necessary, but in practice, finite-time performance is also essential. Both methods often require user specified input parameters that must be selected, and the finite-time performance has a strong dependence on those chosen parameters. Take for example the widely-known finite difference gradient estimate, which requires a perturbation size c. If the function is deterministic, a smaller c would increase the accuracy of the gradient estimate. However, in the stochastic setting, where the function evaluations are noisy, perturbations which are too small can result in extremely noisy gradient estimates. "Too small" is also relative to the problem at hand. The input parameter c has a significant impact on the quality of the estimate and must be chosen appropriately. Ideally, algorithms/methods are robust, and their performance should not be strongly correlated with user specified parameters.

1.1 Motivating Examples

For motivation, we provide application examples where stochastic approximation and sensitivity analysis can be applied, two of which are more detailed.

Queueing networks. Many application areas such as communication networks, production systems, and smartgrids can be modeled using queueing networks. A possible metric of interest is the total throughput of the system. Each node or server has a particular mean service time, which is a controllable parameter, and the

objective is to find the optimal parameter values to maximize the total throughput. Furthermore, the sensitivity of the throughput to the mean service time also be examined.

Inventory management. Production/manufacturing/retail businesses require inventory management systems to help maintain operations and increase profitability. A commonly known inventory policy is the (s, S) policy, where an order is placed to replenish the supply up to S after the inventory on hand falls below s. The goal is to minimize the total cost (e.g., ordering, holding, and shortage) for implementing such a policy.

Finance. Well-modeled stock prices are essential in making well-informed investment decisions. Recently, stochastic volatility models have been introduced with volatility varying in time. The parameters are estimated based on the maximum likelihood function, where SA can be easily applied. Furthermore, payoffs for options are based on stock price models, and optimal exercise strategies can be created to maximize the payoff.

Machine learning using "big data". Traditional regression models are based on batch data, where the analysis is conducted on a fixed data set. However, with the rise in the amount of data available, models must be generated/updated in an online manner, and stochastic approximation is a commonly applied method.

1.1.1 Stochastic Approximation

Consider a G/G/1 queue with one server, unlimited buffer or waiting capacity, interarrival times and service times that follow general distributions with means $1/\lambda$ and x, respectively. In this system, there is only one service station with one server. The customers arrive from outside of the system with interarrival times following some general distribution with mean $1/\lambda$, and the server serves each customer on a first-come first-served basis with a service time also following a general distribution with mean x. This queueing system can be modeled easily with distributional information about the interarrival and service times, and SA can be directly applied. An important objective to consider is increasing customer satisfaction. Among the various correlated performance measures that can be used to quantify the satisfaction level, the total time spent in the system (i.e., time in waiting in queue plus the actual service time) is arguably one of the most significant factors. The objective function could be the expected time in system $E[T(x,\xi)]$, where $T(\cdot,\cdot)$ is the time in system of a customer, plus a positive decreasing cost $C(x)$ associated with the mean service time x, e.g., c/x, where $c > 0$. The objective of stochastic approximation is to find $x^* = \arg\min_{x\in(0,1/\lambda)} E[T(x,\xi)] + C(x)$, by solving $\nabla(E[T(x,\xi)] + c/x) = 0$ iteratively. One could also be interested in the sensitivity of the function, but we will provide a different example for sensitivity analysis.

1.1.2 Sensitivity Analysis

Financial options are instruments that give option holders the right to purchase or sell stock for a particular strike price once certain conditions are met. Investors often use options to hedge financial risk and are especially interested in the expected payoff $E[g(S)]$, where g is the sample payoff and S is the stock price over a time period, which depend on market parameters such as interest rate, initial stock price, volatility, etc., as well as its sensitivity. The derivatives of the expected payoff with respect to market parameters $\partial E[g(S)]/\partial x$ are referred to as Greeks. To implement Greek estimators, the stock price can be modeled as a stochastic process (e.g., Brownian motion and OrnsteinUhlenbeck process) over a pre-specific duration with certain parameters. When making financial decisions, it is essential that investors make them with accurate and reliable information, such as good Greek estimates.

1.2 Contributions

First, we propose two novel stochastic gradient estimation methods in stochastic approximation for univariate and multivariate problems. We consider a setting where direct gradients are also available, so we combine both direct and indirect gradient estimates using a convex weight to form a hybrid gradient. Currently, the existing SA algorithms only consider either direct or indirect gradient estimates, but not in conjunction. For the indirect gradient, we consider a symmetric difference-type gradient estimate, and an average of the two associated direct gradients for the direct gradient estimate. In the one-dimensional case, we use a symmetric dif-

ference gradient estimate, which can be directly extended to higher dimensions, but instead, we employ the well-known simultaneous perturbation gradient to exploit its potential computational efficiency. A critical component of our hybrid gradient estimate is the convex weight, which we derived to minimize the variance of the hybrid gradient, leading to favorable theoretical properties. Our new hybrid gradients are provably convergent in SA.

Second, we conduct preliminary experiments to investigate the sensitivity of the KW algorithm and two of its variants to the step size parameter and explore the tightness of a finite-time MSE bound. Then, we introduce two new adaptive step size sequences for SA, both of which adjust based on the perceived proximity of the current iterate to the optimum. Most of the present adaptive step sizes only consider one adaptive sequence for either each dimension or for all dimensions. Instead, our adaptive step size is based on two sequences, and the step size used in the current iteration depends on the current sample performance(s) compared to the past observations. Although the method is adaptive, the two initial sequences are deterministic, so it also suffers from the same disadvantages of deterministic step sizes. Therefore, we introduce a method to adaptively adjust the two deterministic sequences.

Finally, we examine the sensitivity of two methods: 1) a modified pathwise method involving a kernel estimator to the smoothing bandwidth parameter and 2) a bandwidth selection method to various input parameters.

1.3 Outline

The rest of the dissertation is as follows. In Chapter 2, we provide background information to better understand stochastic approximation and sensitivity analysis. In Section 2.1, we discuss the most common stochastic gradient estimation methods for both direct and indirect gradients. Then in Section 2.2, we introduce the classical SA algorithms in addition to modifications to the gradient, step size, output, and projection operator. Chapter 3 presents our two novel approaches to stochastic gradient estimation methods for single and multidimensional problems along with theoretical and numerical results. We derive optimal weight parameters and prove convergence for both cases. We test our new algorithms against well-known SA algorithms on a deterministic problem with added noise and on a queueing network. In Chapter 4, we explore step size selection techniques in SA, in addition to the tightness of a finite-time MSE bound, before introducing our new adaptive step sizes. We investigate the performance of our new adaptive step sizes on two contrasting deterministic functions with added noise for the one-dimensional case and on a queueing network for the multidimensional case. Chapter 5 presents a proposed modified pathwise gradient estimation method that incorporates a kernel estimator for options with discontinuous payoff functions. We empirically test the sensitivity of the kernel to the bandwidth parameter as well as the sensitivity of a proposed bandwidth selection algorithm to its input parameters. At the end of Chapters 3, 4, and 5, we provide concluding summaries as well as future research directions.

Chapter 2

Background

2.1 Stochastic Gradient Estimation Methods

Stochastic gradient estimation is essential in stochastic approximation and sensitivity analysis but serves different purposes. In SA, the gradient estimate is only an intermediary step that provides a trajectory direction, so it is more tolerant of less accurate estimates. Sensitivity analysis accesses the parameter effects on the function of interest, so the goal is to accurately estimate the gradient, and accuracy is key. Certain stochastic gradient methods do not provide estimates accurate enough to be suitable for sensitivity analysis and are designed specifically for SA.

In general, stochastic gradient estimates fall under one of two main categories: **direct** or **indirect**. Indirect gradient methods *approximate* gradients using finite difference-type gradient estimates, which only require function evaluations, e.g., via the secant method in the one-dimensional case. Since indirect gradient estimates only require sample performances, they can be easily applied to any case but are biased. Direct gradient methods are known as gradient-based approaches, mainly because the techniques results in unbiased gradient estimates, which lead to faster convergence rates in SA.

2.1.1 Indirect Gradients

Indirect gradients are applicable as long as the simulator can generate sample performances. Unfortunately, indirect gradient estimates are biased, which lead to slower asymptotic convergence rates in stochastic approximation.

2.1.1.1 Finite Differences

Finite difference methods stem from Taylor's series expansion, and require the additional task of selecting a perturbation sequence $\{c_n\}$, which impacts the variance. In the one-dimensional case, the derivative can be rewritten as a finite difference plus a bias term that diminishes as the perturbation size c_n approaches zero as $n \to \infty$ (e.g., $f'(x_n) = [f(x_n + c_n) - f(x_n)]/c_n + O(c_n)$). The perturbation size c_n influences the noise level of the finite difference gradient estimate, and if it is too small, the gradient estimates can be very noisy in the stochastic setting. If sensitivity analysis is the ultimate goal, then a relatively larger perturbation size is preferable. In higher dimensions, the idea is to slightly perturb one component while keeping all others constant and return a corresponding function value estimate.

Here are two common examples of finite differences:

$$\widehat{\nabla} J_i(\mathbf{x}_n) = \begin{cases} \dfrac{Y(\mathbf{x}_n + c_n \mathbf{e}_i, \xi^+_{n,i}) - Y(\mathbf{x}_n - c_n \mathbf{e}_i, \xi^-_{n,i})}{2c_n} & \text{symmetric difference,} \\ \dfrac{Y(\mathbf{x}_n + c_n \mathbf{e}_i, \xi^+_{n,i}) - Y(\mathbf{x}_n, \xi_n)}{c_n} & \text{forward difference,} \end{cases}$$

for $i = 1, \ldots, d$, where $\widehat{\nabla} J_i(\mathbf{x}_n)$ is the ith component of the gradient estimate, $\mathbf{e}_i$ denotes the ith unit basis vector, $c_n \in \mathbb{R}^+$ is the perturbation size, $\xi^{\pm}_{n,i}$ and $\xi_{n,i}$ denote the stochastic effects, and $Y(\mathbf{x}, \xi)$ is an unbiased estimate of $J(\mathbf{x})$. Symmetric

differences require the estimate of two function values $J(\mathbf{x}_n + c_n\mathbf{e}_i)$ and $J(\mathbf{x}_n - c_n\mathbf{e}_i)$ for each dimension i and one-sided forward differences require $J(\mathbf{x}_n)$ and $J(\mathbf{x}_n - c_n\mathbf{e}_i)$ for $i = 1, \ldots, d$; therefore, two-sided symmetric differences and one-sided forward difference estimates involve $2d$ and $d+1$ simulation replications, respectively. Since the cost to generate each gradient is linear in the number of dimensions d, finite differences can be very inefficient in higher dimensions. The next indirect gradient estimate addresses this issue.

2.1.1.2 Simultaneous Perturbation

In 1992, [55] introduced the simultaneous perturbation (SP) gradient estimate specifically for multidimensional stochastic approximation problems. Similar to finite differences, SP is a gradient-free approach that only requires objective function values to approximate the underlying gradient, and is therefore easy to implement. The ith component of the SP gradient has the form

$$\widehat{\nabla} J_i(\mathbf{x}_n) = \frac{Y(\mathbf{x}_n + c_n\boldsymbol{\Delta}_n, \xi_n^+) - Y(\mathbf{x}_n - c_n\boldsymbol{\Delta}_n, \xi_n^-)}{2c_n\Delta_{n,i}},$$

where $c_n \in \mathbb{R}^+$ is the perturbation size, $\xi_n^\pm$ denotes the stochastic effects, $\boldsymbol{\Delta}_n = (\Delta_{n,1}, \ldots, \Delta_{n,d})$, and the sequence $\{\boldsymbol{\Delta}_n\}$ is generally assumed to be i.i.d. and independent across components. Notice that, unlike finite differences, the numerator of the SP gradient is independent of i, so a gradient only requires 2 simulation runs, regardless of the number of dimensions d, which can be efficient in higher dimensions. The only additional requirement is to generate $\boldsymbol{\Delta}_n$, which is relatively inexpensive compared to simulating sample performances. The SP gradient is only intended for

use in stochastic approximation algorithms and has limited relevance in sensitivity analysis.

2.1.1.3 Random Directions

The random directions (RD) gradient was also developed specifically for multidimensional problems and was the inspiration behind SP. The ith component of the RD gradient is

$$\widehat{\nabla} J_i(\mathbf{x}_n) = \frac{\left[Y(\mathbf{x}_n + c_n \boldsymbol{\Delta}_n, \xi_n^+) - Y(\mathbf{x}_n - c_n \boldsymbol{\Delta}_n, \xi_n^-)\right] \Delta_{n,i}}{2c_n},$$

with perturbation size $c_n \in \mathbb{R}^+$ and stochastic perturbation $\boldsymbol{\Delta}_n = (\Delta_{n,1}, \ldots, \Delta_{n,d})$. The sequence $\{\boldsymbol{\Delta}_n\}$ is generally assumed to be i.i.d. and independent across components, and $\xi_n^{\pm}$ denote the stochastic effects. Notice that RD is almost identical to SP, except instead of dividing by the stochastic perturbation component $\Delta_{n,i}$, it multiplies the difference term. This difference changes the restrictions on the stochastic sequence $\{\boldsymbol{\Delta}_n\}$ to guarentee convergence in SA and translates to a bound on the second moment, instead of the inverse moment, with zero mean, so the Gaussian distribution is applicable [23].

2.1.2 Direct Gradients

Direct gradients generally require more "offline" work and involve additional coding within the simulation model. However, if direct gradient methods are applicable, they usually provide unbiased gradient estimates, which lead to faster asymptotic convergence rates in stochastic approximation. In addition, direct gradient

methods are computationally efficient, especially in higher dimensions, since they often only require one simulation replication for any dimension d.

For notation, assume

$$J(X) = E[Y(\theta)] = E[Y(X_1, \ldots, X_d)], \tag{2.1}$$

where $Y : \mathbb{R}^d \to \mathbb{R}$ and $X = (X_1, \ldots, X_d)$ is a vector of input random variables. The parameter θ has been purposely left out of the right-hand side of expression (2.1) to emphasize the point that it can appear in either the sample (pathwise) or measure (distributional). Let f denote the joint distribution of the input random variables X, then (2.1) can be written in the following two ways:

$$E[Y(X)] = \begin{cases} \displaystyle\int_{-\infty}^{\infty} Y(x) f(x;\theta) dx & \text{distributional,} \\ \displaystyle\int_0^1 Y(X(\theta;u)) du & \text{pathwise,} \end{cases} \tag{2.2}$$

where x, u, and the integrals are d-dimensional. When the parameter is located in $Y(\cdot)$, then the dependency of θ is pathwise and perturbation analysis is usually applied; whereas, if θ is located in $f(\cdot)$, then its θ dependency is distributional and the likelihood ratio or score function and weak derivatives could potentially be applied. In some instances, the θ can be pushed in and/or out of the distribution with an appropriate change of variables, also known as the push in/out method, which will change the applicability of the estimation method.

If we assume that the interchange of the integration and differentiation is valid, then differentiating (2.2) with respect to θ results in the following two cases:

$$\frac{dE[Y(X)]}{d\theta} = \begin{cases} \displaystyle\int_{-\infty}^{\infty} Y(x) \frac{df(x;\theta)}{d\theta} dx & \text{distributional,} \\ \displaystyle\int_0^1 \frac{dY(X(\theta;u))}{d\theta} du & \text{pathwise.} \end{cases} \tag{2.3}$$

We assume each random number u_i produces a random variate x_i for $i = 1, \ldots, d$.

For simplicity, in Sections 2.1.2.1 and 2.1.2.2, we assume that the parameter only appears in X_1, which is generated independently of the other random variables. For the simplified example, we will apply the method to the specific case where $d = 2$, $X_1 \sim \exp(\theta)$, and $X_2 \sim U(0, 1)$. The random variate x_1 is generated via inverse transform method where $X_i = -\theta \ln U_i$ and $U_i \sim U(0, 1)$.

2.1.2.1 Infinitesimal Perturbation Analysis

For the pathwise dependent case, the second integral can be expressed as

$$\begin{aligned} \frac{\partial E[Y(X)]}{\partial \theta} &= \int_0^1 \frac{dY(X_1(\theta; u_1), X_2, \ldots, X_d)}{d\theta} du \\ &= \int_0^1 \frac{\partial Y(X)}{\partial X_1} \frac{dX_1(\theta; u)}{d\theta} du, \end{aligned}$$

where $\frac{\partial Y}{\partial X_1} \frac{\partial X_1}{\partial \theta}$ is the infinitesimal perturbation analysis (IPA) estimator.

In this particular case,

$$\frac{\partial E[Y(X)]}{\partial \theta} = \int_0^1 \int_0^1 \frac{\partial Y(X_1(\theta; u_1), u_2)}{\partial x_1} \frac{dX_1(u_1; \theta)}{d\theta} du_1 du_2,$$

and $\frac{dX_1}{d\theta}$ depends on the construction of X_1 and since $X_1 = -\theta \ln U$, then $\frac{dX_1}{d\theta} = -\ln U = \frac{X_1}{\theta}$, so the final IPA estimator is

$$\frac{\partial Y(X_1, X_2)}{\partial X_1} \frac{dX_1}{d\theta} = \frac{\partial Y(X_1, X_2)}{\partial X_1} \frac{X_1}{\theta}.$$

Implementation of the estimator requires more knowledge about how the X_i's are generated in order to compute the derivative of the random variable $\frac{dX_i}{d\theta}$; therefore the representation of X_i, which depends on θ, is key. As a general rule, if the

sample performance is continuous with respect to the parameter of interest, then IPA is a suitable gradient estimation method. However, in practice, discontinuities may occur in the sample performance (e.g., indicator function), thereby forcing other estimation techniques to be applied. Refer to [23, 25] for detailed examples.

The next two types of estimators pertain to the distribution dependent case.

2.1.2.2 Likelihood Ratio/Score Function

Assume $f_i(\cdot;\theta)$ is the marginal p.d.f. of X_i, so the joint density can be written as $f(x) = \Pi_{i=1}^{d} f_i(x_i)$. Therefore, (2.3) can be expressed as:

$$\begin{aligned}\frac{dE[Y(X)]}{d\theta} &= \int_{-\infty}^{\infty} Y(x)\frac{\partial f_1(x_1;\theta)}{\partial\theta}\Pi_{k=2}^{d} f_k(x_k)dx \\ &= \int_{-\infty}^{\infty} Y(x)\frac{\partial \ln f_1(x_1;\theta)}{\partial\theta} f(x)dx,\end{aligned}$$

where $Y(X)\frac{\partial \ln f_1(X_1;\theta)}{\partial\theta}$ is the score function or likelihood ratio estimator.

In our particular example,

$$\begin{aligned}\frac{dE[Y(X)]}{d\theta} &= \int_0^1\int_0^{\infty} Y(x_1,x_2)\frac{\partial f_1(x_1;\theta)}{\partial\theta}dx_1dx_2 \\ &= \int_0^1\int_0^{\infty} Y(x_1,x_2)\left[\frac{1}{\theta}\left(\frac{x_1}{\theta}-1\right)\right]\frac{1}{\theta}e^{-\frac{x_1}{\theta}}dx_1dx_2.\end{aligned}$$

SF/LR estimators are generally simple to implement when they are applicable. However, this method is not applicable to distributions where the underlying support depends on θ. For instance, $U(0,\theta)$ and $Ber(p;\theta,b)$ are examples of continuous and discrete distributions, respectively, with θ dependent support, so LR/SF estimators do not exist. However, they exist for distributions where the underlying probabilities rely on the parameter of interest, such as $bin(n,\theta)$, $Ber(\theta;a,b)$, $Poisson(\theta)$, $exp(\theta)$,

and $N(\theta, \sigma^2)$. Furthermore, this method usually has difficulty with nondistributional parameters, but the issue might be overcome by using the push in or push out method to push the parameter of interest in or out of the distribution. If the parameter appears in more than one input random variable distribution, the variance will increase linearly as the number of times those random variables are used in the simulation, but this issue could be controlled through batching. For example, if the performance measure of interest of an M/M/1 queue is the average time in system of 1000 customers, we could look at the average of 100 samples, 10 customers each, instead. Furthermore, for higher derivative estimators, this method is easier to apply.

2.1.2.3 Weak Derivatives

For the weak derivatives estimator, there exists $c(\theta)$ and densities $f_1^{(2)}$ and $f_1^{(1)}$ such that

$$\frac{\partial f_1}{\partial \theta} = c(\theta)[f_1^{(2)} - f_1^{(1)}]. \tag{2.4}$$

This representation always exists, because if we let $f_1^{(1)} = \frac{1}{c}\left(\frac{\partial f_1}{\partial \theta}\right)^-$ and $f_1^{(2)} = \frac{1}{c}\left(\frac{\partial f_1}{\partial \theta}\right)^+$, where $c = \int \left(\frac{\partial f_1}{\partial \theta}\right)^- dx = \int \left(\frac{\partial f_1}{\partial \theta}\right)^+ dx$, then $\frac{\partial f_1}{\partial \theta}$ is written as the difference of two signed measures, otherwise known as the Hahn-Jordan decomposition, which is not unique.

Therefore, using this representation,

$$\begin{aligned} \frac{dE[Y(X)]}{d\theta} &= \int_{-\infty}^{\infty} Y(x)\frac{\partial f(x;\theta)}{\partial \theta}dx \\ &= \int_{-\infty}^{\infty} Y(x)c(\theta)[f_1^{(2)} - f_1^{(1)}]dx, \end{aligned}$$

where the weak derivatives estimator is of the form

$$c(\theta)[Y(X_1^{(2)}, X_2, \ldots, X_d) - Y(X_1^{(1)}, X_2, \ldots, X_d)],$$

where $X_1^{(1)} \sim f_1^{(1)}$ and $X_1^{(2)} \sim f_1^{(2)}$.

This is called a weak derivative because the left hand side of (2.4) might not be proper, but when it is integrated against a test function, it is well-defined. These estimators are not unique and in our specific example,

$$\frac{dE[Y(X)]}{d\theta} = \frac{1}{\theta}\int_0^1 \int_0^\infty Y(x_1, x_2)\left[\frac{1}{\theta}\left(\frac{x_1}{\theta} - 1\right)\right]\frac{1}{\theta}e^{-\frac{x_1}{\theta}}dx_1 dx_2,$$

where the weak derivatives estimator is $\frac{1}{\theta}[Y(X_1^{(2)}, X_2) - Y(X_1^{(1)}, X_2)]$, with $X_1^{(2)} \sim Erl(2, \theta)$ and $X_1^{(1)} \sim \exp(\theta)$.

Similar to the LR method, WD estimators variance grows linearly as the number of random variables with θ dependency increases in the simulation. For example, in a queueing system, θ could appear in the interarrival and service times, which can be decreased through batching. In addition, WDs are not unique, so deciding which to use in generating the estimator could be a challenge.

2.1.2.4 General Extension

We now generalize the estimators for situations where more X_i's have a dependence on θ. Let P^* denote the set of indices where X_i is dependent on θ. Then the following are the general estimators:

IPA estimator:

$$\sum_{i \in P^*} \frac{\partial Y(X)}{\partial X_i}\frac{dX_i}{d\theta}$$

LR/SF estimator (multivariate, independent):

$$Y(X)\frac{\partial \ln f(X;\theta)}{\partial \theta}, Y(X)\sum_{i\in P^*}\frac{\partial \ln f_i(X_i;\theta)}{\partial \theta}$$

WD estimator (multivariate, independent):

$$c(\theta)(L(X^{(2)}) - L(X^{(1)})), \sum_{i\in P^*} c_i(\theta)(L(X_1\ldots, X_i^{(2)}, \ldots, X_d) - L(X_1\ldots, X_i^{(1)}, \ldots, X_d))$$

2.2 Stochastic Approximation

2.2.1 Classical Methods

The two classical stochastic approximation methods, Robbins-Monro (RM) and Kiefer-Wolfowitz (KW), were first applied to the one-dimensional unconstrained stochastic optimization problem, and the recursive scheme follows

$$x_{n+1} = x_n - a_n\widehat{\nabla} J(x_n), \tag{2.5}$$

which is identical to (1.2) with the exception of the projection operator. The main difference between RM an KW is the gradient estimate $\widehat{\nabla} J$. In RM, the gradient is estimated by an unbiased estimator, whereas in KW, the gradient estimate is only asymptotically unbiased. Both algorithms have their advantages and will be discussed in the next two sections.

2.2.1.1 Robbins-Monro

Robbins and Monro pioneered the way for stochastic approximation in [50], and the number of literature citations since has grown to over 3500. RM was first

intended to solve root-finding problems, i.e., $h(\mathbf{x}^*) = 0$, where $h : \mathbb{R}^d \to \mathbb{R}$. The Robbins-Monro (RM) stochastic approximation algorithm was applied to a stochastic optimization problem with the objective function J by setting $h = \nabla J$, where the true gradient is estimated using an unbiased direct gradient. RM solves this problem iteratively as in (2.5) by choosing the gradient estimate $\widehat{\nabla} J(x_n)$ to be an unbiased estimator, i.e., $E[\widehat{\nabla} J(x_n)] = \nabla J(x_n)$, and the output is taken as the last iterate x_N^*, where N is the stopping time. Unfortunately, the direct gradient measurements are still approximations to the actual gradients because of the noise (i.e., $\widehat{\nabla} J(x_n) = \nabla J(x_n) + \delta_n$, where ϵ_n is noise with zero mean). When used in SA algorithms, unbiased gradients estimates lead to faster convergence rates, and under certain conditions RM can achieve an asymptotic convergence rate of up to $O(n^{-1/2})$ [53]. The following is the original convergence theorem.

Theorem 2.2.1. *(Theorem 2 [50]) Assume $\nabla J(x)$ has a unique root x^* and suppose $\widehat{\nabla} J(x)$ is an unbiased gradient estimator, i.e., $E[\widehat{\nabla} J(x)] = \nabla J(x)$. If the sequence $\{x_n\}$ is generated from (2.5) and the following conditions hold:*

1. *$\{a_n\}$ is a sequence of positive constants such that $\sum_{n=1}^{\infty} a_n = \infty$, $\sum_{n=1}^{\infty} a_n^2 < \infty$.*

2. *$\nabla J(x) \geq 0$ for $x > x^*$ and $\nabla J(x) \leq 0$ for $x < x^*$.*

3. *There exists a positive constant C such that $P(|\widehat{\nabla} J(x)| \leq C) = 1 \ \forall x$.*

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

The objective function J is assumed to have a global minimum with a bounded

derivative. The most well-known conditions are restrictions on the gain sequence $\{a_n\}$. Generally, the step size $a_n \to 0$ 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 towards x^* and not get stuck at a poor estimate. The typical form for the step size is $a_n = \theta_a/(n + A)^\alpha$, 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 restrictions still allow for an uncountable number of step size options, and the finite-time performance of SA is notoriously known to be sensitive to the a_n choice. Theoretically, unbiased gradients lead to faster convergence rates but are not always available, so the next method addresses this issue.

2.2.1.2 Kiefer-Wolfowitz

The Kiefer-Wolfowitz stochastic approximation algorithm only requires sample performances measurements to implement and does not require additional information on the system dynamics or input distributions as in RM. The original KW iterative scheme

$$x_{n+1} = x_n - a_n \frac{Y(x_n + c_n, \xi_n^+) - Y(x_n - c_n, \xi_n^-)}{2c_n}, \tag{2.6}$$

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 [38].

Theorem 2.2.2. *([37]) Assume $J(x) = E[Y(x,\xi)]$. If the sequence $\{x_n\}$ is generated from (2.6) and the following conditions hold:*

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

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

2. *$J(x)$ is strictly decreasing for $x < x^*$ and strictly increasing for $x > x^*$.*

3. *$Var(Y(x,\xi)) < \infty$ and satisfies the following regularity conditions:*

   1) There exist positive constants β and B such that

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

   2) There exist positive ρ and R such that

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

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

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

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 prohibits the finite difference step size c_n from decreasing too quickly, as well as to prevent noisy gradients. The second condition insures that there is a global optimum. The first regularity condition requires $J(x)$ to be locally Lipschitz in a neighborhood of

x^*; the second one prevents $J(x)$ 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 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 as discussed in Section 2.1.1.1. 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 gradient estimates. Even though both schemes are easy to implement, their convergence rates are typically inferior to the RM algorithm, although under certain conditions with CRN $\xi^+_{n,i} = \xi^-_{n,i}$ for the symmetric difference, they 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 perturbation sequence $\{c_n\}$. In general, KW is a simple algorithm to implement for simulation optimization applications, albeit costly in high-dimensional settings.

2.2.2 Robust Gradient

2.2.2.1 Simultaneous Perturbation Stochastic Approximation

Simultaneous perturbation stochastic approximation (SPSA) specifically addresses multivariate optimization problems [55]. Let ϵ_n^+ and ϵ_n^- be the noise from the sample performances $J(\mathbf{x}_n + c_n\mathbf{\Delta}_n)$ and $J(\mathbf{x}_n - c_n\mathbf{\Delta}_n)$, respectively.

SPSA Algorithm

- Input. Choose $\mathbf{x}_1 \in \Theta$, $\{a_n\}$, $\{c_n\}$, and stopping time N.
- Initialize. Let $n = 1$.
- While $n < N$,
  - Step 1. Generate a d-dimensional random perturbation vector $\mathbf{\Delta}_n$.
  - Step 2. Generate an estimate of $\nabla J(\mathbf{x}_n)$:

$$\widehat{\nabla} J(\mathbf{x}_n) = \frac{Y(\mathbf{x}_n + c_n\mathbf{\Delta}_n, \xi_n^+) - Y(\mathbf{x}_n - c_n\mathbf{\Delta}_n, \xi_n^-)}{2c_n} \begin{bmatrix} \Delta_{n,1}^{-1} \\ \vdots \\ \Delta_{n,d}^{-1} \end{bmatrix}$$

  - Step 3. Compute $\mathbf{x}_{n+1} = \mathbf{x}_n - a_n\widehat{\nabla} J(\mathbf{x}_n)$.
  - Step 4. Let $n = n + 1$. Go to Step 1.
- Output. $\mathbf{x}_N^* = \mathbf{x}_N$.

Theorem 2.2.3. *(Theorem 7.1 [57]) Suppose J has a unique minimum $\mathbf{x}^* \in \Theta$ and $\{\mathbf{x}_n\}$ 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 $\sum_{n=0}^{\infty} a_n = \infty$ and $\sum_{n=0}^{\infty} a_n^2 c_n^{-2} < \infty$.*

2. *The function $J(\mathbf{x}) \in C^3$ and bounded on $\mathbb{R}^d$.*

3. *$||\mathbf{x}_n|| < \infty$ for all n.*

4. *$E[\epsilon_n^+ - \epsilon_n^- | \mathbf{\Delta}_n, \mathcal{F}_n] = 0$ and $E[(Y(\mathbf{x}_n \pm c_n \mathbf{\Delta}_n, \xi_n^{\pm})/\Delta_{n,i})^2]$ is uniformly bounded for all n, i.*

5. *$\mathbf{x}^*$ is an asymptotically stable solution of the differential equation $\partial \mathbf{x}(t)/\partial t = -\nabla J(\mathbf{x}(t))$.*

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 $\mathbf{x}_n \to \mathbf{x}^*$ a.s. as $n \to \infty$.

The optimal convergence rate for SPSA is $O(n^{-1/3})$ [55]. Various convergence proofs have been presented with slight modifications to the conditions (cf. [12, 17, 30, 55, 60]). The perturbation sequence $\{\mathbf{\Delta}_n\}$ where $\mathbf{\Delta}_n = (\Delta_{n,1}, \ldots, \Delta_{n,d})$ with the sequence $\{\Delta_{n,i}\}$ is independent with mean zero and finite inverse moments (i.e., $E[\Delta_{n,i}] = 0$ and $E[|\Delta_{n,i}|^{-1}] < \infty$ for $i = 1, \ldots, d$) to guarantee a.s. convergence when applied to SA. 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 (e.g., ± 1) each with probability 1/2. In addition, an appropriately scaled $\mathbf{x}_n$ is approximately normal for large n, and the relative efficiency of SPSA depends on the geometric shape of $J(\mathbf{x})$, choice of $\{a_n\}$, $\{c_n\}$, distribution of $\{\Delta_{n,i}\}$, and noise level.

Many extensions to the original SPSA algorithm have been developed, for example, the constrained setting using projection operators [26, 54]. A slight modification is the averaging of the SPSA gradient estimators, where 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 $\nabla^2 J(x)$ to accelerate the convergence [57], analogous to the Newton-Raphson method. Iterate averaging in the SPSA setting has also been explored, but performs relatively poor in finite-time [17, 56]. 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.

2.2.2.2 Gradient Averaging

Gradient averaging can help stabilize the gradient estimate, especially if it is noisy. One obvious straightforward method is to generate multiple gradient estimates at each iteration and average them to provide a better estimate, but this process can be expensive and is not worth the computational effort, according to

Robbins and Monro. In SA, the gradient estimate is only an intermediate step that provides a direction for the iterates to move, but the goal is to find the optimum $\mathbf{x}^*$ and not the best estimate of the gradient as in sensitivity analysis. The computational cost could be better expended in future iterations. Another form of gradient averaging uses previous gradient estimates in the current gradient approximation. One of the earliest proposed gradient averaging methods was introduced in [22] and [27], where "present" and "past" data is used to improve convergence properties of SA. The method introduced in [22] considers all past gradients with the gradient update of the form

$$\mathbf{d}_{n+1} = \mathbf{d}_n + b_n(\widehat{\nabla} J(\mathbf{x}_n) - \mathbf{d}_n),$$

where $\mathbf{d}_n$ represents the previous direction, $\widehat{\nabla} J(\mathbf{x}_n, \xi_n)$ is the new gradient, b_n is the averaging coefficient, and $\mathbf{d}_{n+1}$ is the new updated direction used in the stochastic approximation algorithm. Later, [27] proposed a modified version of this gradient averaging technique, which involves an additional step.

$$\begin{aligned} \mathbf{d}_{n+1} &= \underline{\mathbf{d}}_n + b_n(\widehat{\nabla} J(\mathbf{x}_n) - \underline{\mathbf{d}}_n), \\ \underline{\mathbf{d}}_n &= \mathbf{d}_n + T(\mathbf{x}_n, \boldsymbol{\Delta}_n), \end{aligned}$$

where $\boldsymbol{\Delta}_n = \mathbf{x}_n - \mathbf{x}_{n-1}$, $T(\mathbf{x}_n, \boldsymbol{\Delta}_n)$ represents a updating function with higher derivatives, and $\mathbf{d}_{n+1}$ is the new gradient estimate used in SA, which incorporates higher derivatives. In addition, [27] propose an updating method that averages past gradient estimates if the change in estimates does not surpass a certain value.

2.2.3 Adaptive Step Sizes

2.2.3.1 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 [36] 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 $x_1 \in \Theta, \{a_n\}$, Π_Θ, and stopping time N.
- Initialize.
  - Let $n = 2$ and $k = 1$.

- Generate an estimate $\widehat{\nabla} J(x_1)$ of $\nabla J(x_1)$.
- Compute $x_2 = \Pi_\Theta(x_1 - a_1 \widehat{\nabla} J(x_1))$.

- While $n < N$,
  - Step 1. Generate an estimate $\widehat{\nabla} J(x_n)$ of $\nabla J(x_n)$.
  - Step 2. Compute $x_{n+1} = \Pi_\Theta(x_n - a_k \widehat{\nabla} J(x_n))$. 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 Theorem 2.2.1 and 2.2.2 for RM and KW, respectively [36]. An extension of Kesten's rule to higher dimensions is discussed in [15]. See [29] for an extensive review of both deterministic and stochastic step sizes.

2.2.3.2 Scaled-and-Shifted Kiefer-Wolfowitz

The scaled-and-shifted Kiefer-Wolfowitz (SSKW) algorithm [7] 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. $\{a_n\}$, $\{c_n\}$, $\Theta = [l, u]$, Π_Θ, stopping time N, and
  - h_0 = number of forced boundary hits,
  - γ_0 = scale up factor for $\{c_n\}$,
  - k_a = maximum number of shifts of $\{a_n\}$,
  - v_a = initial upper bound of shift,
  - ϕ_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 $\widehat{\nabla} J(x_n)$ using symmetric differences.

    * Compute $x_{n+1} = \Pi_\Theta \left(x_n - a_n \widehat{\nabla} J(x_n) \right)$.

      · 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 $\widehat{\nabla} J(x_n)$ using symmetric differences.
    - Compute x_{n+1} using (1.2).
      - 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 $sh \leq k_a$.
      - 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 $sc \leq k_c$.
      - Otherwise, go to Step 4.
  - Step 2.
    - Find smallest integer β' 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 stops 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 α, 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 (β) 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 γ, 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 the sequences by a constant and the shifting phase only scales up the $\{c_n\}$ finitely many times and skips a finite number of terms in $\{a_n\}$.

2.2.4 Robust Output

2.2.4.1 Averaging Iterates

Iterate averaging, introduced in [51] and [49], 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 that the iterates surround the optimum in a balanced manner, and that the domain in which the iterates oscillate shrinks as n increases. Averaging trajectories reduces the sensitivity to the initial step size choice. The algorithm follows recursion (1.2) for the RM case; however, instead of the taking the last iterate $\mathbf{x}_N$ as the output,

the optimum is estimated by

$$\mathbf{x}_N^* = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n,$$

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, $\sqrt{n}(\mathbf{x}_n^* - \mathbf{x}^*)$ is asymptotically normal with mean zero and the smallest covariance matrix, which is the inverse of the average Fisher information matrix. (cf. [49]). A constant step size can be applied and yields convergence in distribution [42].

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

$$\mathbf{x}_N^* = \frac{1}{m}\sum_{n=N-m+1}^{N}\mathbf{x}_n. \tag{2.7}$$

An advantage of (2.7) 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 $\mathbf{x}^*$. Asymptotic normality for a growing window is shown in [40] and [42], which also includes constant step sizes. Another modification of the original method incorporates $\mathbf{x}_N$ in the components being averaged $\mathbf{x}_{N-1}^*$, which is known as the feedback approach [41]. These methods are suited for problems where the iterates hover around the optimum. In an empirical study, iterate averaging was applied to SPSA [44]. The results suggest that if the Hessian of $J(\mathbf{x})$ is large, averaging is considered ideal, since it is associated with a high variability in $J(\mathbf{x})$, 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 [17]. Under certain parameter settings, iterate averaging and weighted averaging produce the same estimator.

2.2.4.2 Robust Stochastic Approximation

The robust SA (RSA) method is intended to be relatively insensitive to the choose of the step size sequence, similar to Polyak-Ruppert iterate averaging. The form of RSA is identical to (1.2) with the exception of the output. Instead of $\mathbf{x}_N^* = \mathbf{x}_N$, where $\mathbf{x}_N$ is the last iterate, $\mathbf{x}_N^*$ is calculated as

$$\mathbf{x}_N^* = \frac{\sum_{n=1}^N a_n \mathbf{x}_n}{\sum_{n=1}^N a_n},$$

where $a_n > 0$ for all n. It is clear that if $a_n = a$, where $a \in \mathbb{R}^+$ for all n, then $\mathbf{x}_N^* = \frac{1}{N}\sum_{n=1}^N \mathbf{x}_n$, 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 [42]. A finite-time bound for $E[J(\mathbf{x}_N^*) - J(\mathbf{x})]$ was derived under RSA when J is assumed convex [45]. Assume there exists $C > 0$ such that $E[||\nabla J(\mathbf{x})||] \leq C^2$ for all $\mathbf{x} \in \Theta$. Then for an N-step iteration policy,

$$E[J(\mathbf{x}_N^*) - J(\mathbf{x})] \leq \frac{||\mathbf{x}_0 - \mathbf{x}^*||^2 + C^2 \sum_{n=1}^N a_n^2}{2\sum_{n=1}^N a_n}. \tag{2.8}$$

For equal weights or iterate averaging, the bound on the right hand side of (2.8) can be minimized if

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

where $D_\Theta = \max_{\mathbf{x},\mathbf{y}\in\Theta} ||\mathbf{x} - \mathbf{y}||$. The distance $||\mathbf{x}_0 - \mathbf{x}^*||$ in place of D_Θ tightens the bound in (2.8), but $\mathbf{x}^*$ is unknown so the improvement may not be practically meaningful. The 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

$$\mathbf{x}^*_{N,K} = \frac{\sum_{n=K}^{N} a_n \mathbf{x}_n}{\sum_{n=K}^{N} a_n}.$$

If we consider the varying step size

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

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

$$E[J(\mathbf{x}^*_{N,K}) - J(\mathbf{x})] \leq \frac{D_\Theta C}{\sqrt{N}} \left[\frac{2}{\theta} \left(\frac{N}{N-K+1} \right) \frac{\theta}{2} \sqrt{\frac{N}{K}} \right],$$

for $1 \leq K \leq N$.

2.2.4.3 Acceleration Stochastic Approximation

The Accelerated SA (AC-SA) algorithm [32] 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 $J(\cdot)$ for the convex case.

AC-SA is an example of a proximal method, which introduces a proximity function into the objective function. The prox-function acts as a regularization term to prevent the next iterate update $\mathbf{x}_{n+1}$ from being too far from $\mathbf{x}_n$ 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

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

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

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

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

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

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

for all $\mathbf{x}, \mathbf{y} \in \Theta$ where $\mu > 0$ is the strong convexity parameter. If $M > 0$ and $L = 0$, then J is Lipschitz continuous with Lipschitz constant $M/2$. If $M = 0$ and $L > 0$, then J has Lipschitz continuous gradients with Lipschitz constant L.

The AC-SA algorithm updates three sequences, $\{\mathbf{x}_n^{md}\}$, $\{\mathbf{x}_n^{ag}\}$, and $\{\mathbf{x}_n\}$. 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 $V(\mathbf{x}, \mathbf{y})$, $\{\alpha_n\}$ 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 $\mathbf{x}_0^{ag} = \mathbf{x}_0 \in \Theta$ and let $n = 1$.
- While $n < N$,
  - Step 1. Generate an estimate $\widehat{\nabla} J(\mathbf{x}_n)$ of $\nabla J(\mathbf{x}_n)$.
  - Step 2. Compute

$$\begin{aligned}
\mathbf{x}_n^{md} &= \frac{\alpha_n[(1-\alpha_n)\mu + \gamma_n]}{\gamma_n + (1-\alpha_n^2)\mu}\mathbf{x}_{n-1} + (1-\alpha_n)\frac{(1-\alpha_n)(\mu+\gamma_n)}{\gamma_n + (1-\alpha_n^2)\mu}\mathbf{x}_{n-1}^{ag} \\
\mathbf{x}_n &= \arg\min_{\mathbf{x}\in\Theta}\{\alpha_n[\langle \nabla J(\mathbf{x}_n^{md}), \mathbf{x}\rangle + \mu V(\mathbf{x}_n^{md})] + [(1-\alpha_n)\mu + \gamma_n]V(\mathbf{x}_{n-1}, \mathbf{x})\} \\
\mathbf{x}_n^{ag} &= \alpha_n \mathbf{x}_n + (1-\alpha_n)\mathbf{x}_{n-1}^{ag}
\end{aligned}$$

  - Step 3. Let $n = n + 1$ and go to Step 1.
- Output. $\mathbf{x}_N^* = \mathbf{x}_N^{ag}$.

Note: $V(\mathbf{x}, \mathbf{y}) = \frac{1}{2}||\mathbf{x} - \mathbf{y}||^2$ using the Euclidean norm with $\nu = 1$ is a common prox-function. Refer to [31] for details.

Theorem 2.2.4. *(Theorem 1 [31]) Assume* $V(\mathbf{x},\mathbf{y}) \leq \frac{1}{2}||\mathbf{x}-\mathbf{y}||^2$ *for all* $\mathbf{x},\mathbf{y} \in \Theta$ *when* $\mu > 0$ *and* $E[(\widehat{\nabla} J(\mathbf{x})) - \nabla J(\mathbf{x}))^2] \leq \sigma^2 \ \forall \mathbf{x} \in \Theta$. *Choose* $\{\alpha_n\}$ *and* $\{\gamma_n\}$ *such that*

$$\nu(\mu + \gamma_n) > L\alpha_n^2, \tag{2.9}$$

$$\gamma_n/\Gamma_n = \gamma_{n+1}/\Gamma_{n+1} \text{ for } \ n \geq 1, \tag{2.10}$$

where

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

Then,

$$E[J(\mathbf{x}_N^{ag}) - J(\mathbf{x}^*)] \ \leq \ \Gamma_N \left(\gamma_1 V(\mathbf{x}_0, \mathbf{x}^*) + \sum_{n=1}^{N} \frac{2(M^2+\sigma^2)\alpha_n^2}{\Gamma_n[\nu(\mu+\gamma_n) - L\alpha_n^2]} \right). \tag{2.11}$$

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 (2.9) and (2.10). Under these conditions, the right hand side of (2.12) can be bounded by

$$\frac{4LV(\mathbf{x}_0,\mathbf{x}^*)}{\nu N(N+1)} + \frac{8(M^2+\sigma^2)}{\nu\mu(N+1)}, \tag{2.12}$$

for $\mu > 0$. The bounds in (2.11) and (2.12) rely on additional information of the function and gradient, which are unknown, so they must be approximated.

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 $\mathbf{x}_n^{md}$ and $\mathbf{x}_n$ update since $\mu = 0$. The resulting

updates are

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

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

$$\mathbf{x}_n = \Pi_\Theta \left(\mathbf{x}_{n-1} - \frac{\alpha_n}{\gamma_n} \widehat{\nabla} J(\mathbf{x}_n^{md}) \right), \tag{2.13}$$

which has a similar form to the standard SA algorithm. Notice in the update for $\mathbf{x}_n$ in (2.13), α_n/γ_n takes the place of the step size a_n in (1.2) and the gradient estimate $\widehat{\nabla} J$ is evaluated at $\mathbf{x}_n^{md}$ as opposed to $\mathbf{x}_{n-1}$. If we consider the same parameter setting as in the strongly convex case, the "step size" α_n/γ_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 increases to infinity [31].

Theorem 2.2.5. *(Proposition 7 [31]) Assume that the assumptions in Theorem 3.3.2 hold for $\mu = 0$ and the sequences $\alpha_n = 1/(n+1)$ and $\gamma_n = 4\gamma/[\nu n(n+1)]$ for $\gamma \geq 2L$. Then*

$$E[J(\mathbf{x}_N^{ag}) - J(\mathbf{x}^*)] \leq \frac{4\gamma V(\mathbf{x}_0, \mathbf{x}^*)}{\nu N(N+1)} + \frac{4(M^2+\sigma^2)(N+2)}{3\gamma}, \tag{2.14}$$

where

$$\gamma = \max\left\{ 2L, \left[\frac{\nu(M^2+\sigma^2)N(N+1)(N+2)}{3V(\mathbf{x}_0, \mathbf{x}^*)} \right]^{1/2} \right\}.$$

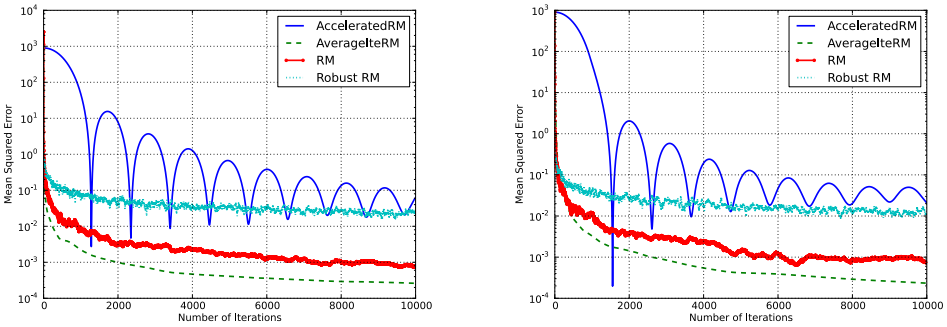
minimizes the bound in (2.14).

2.2.4.4 Numerical Comparison

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 1000 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 (2.8), 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 γ is given in (2.15) with $\nu = 1$, $L = 2/3$ and $M = 0$.

Figure 2.1 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



(a) $[-50, 50]$ **(b)** $[-5, 95]$

Figure 2.1: MSE under AC-SA, RM, RM w/averaging and RSA for $f(x) = -\frac{1}{3}x^2$, $x_1 = 30.0$, $\sigma = 1.0$.

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 $\{\mathbf{x}_n^{ag}\}$ appear to have the same behavior, following a smooth oscillating path/curve. From Figure 2.1, the AC-SA curve appears to level off and hover slightly over the RSA curve. The stopping time dictates 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 have 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 10000 iterations. Both the AC-SA and RSA algorithms require additional knowledge to choose the optimal step size that minimizes the bound in (2.11) and (2.8) for AC-SA and RSA, respectively.

2.2.5 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 $\mathbf{x}^* \in \Theta$ with high probability; however, this may increase the potential of an algorithm to perform poorly due to the size of the parameter search space [2] . 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 thresh-

old, with the threshold increasing after it is exceeded [13]. 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 [64]. 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 Θ as an increasing sequence of compact sets (i.e., $\Theta_n \subseteq \Theta_{n+1}$, where $\Theta = \cup\Theta_n$) was introduced [2]. The orthogonal projection operator changes from Π_{Θ_k} to $\Pi_{\Theta_{k+1}}$ if $\mathbf{x}_{n+1} \notin \Theta_k$. The idea is to start with a smaller feasible region Θ_1 and only increase when there is reason to believe the optimum $\mathbf{x}^* \notin \Theta_1$ (i.e., when the $\mathbf{x}_k \notin \Theta_1$). Since the projection is made onto the current compact set Θ_m, the progress gained up to that point is not lost. The feasible region Θ is written as $\cup_k\Theta_k$, so it is impossible for $\mathbf{x}^* \notin \Theta_k$ for some k. If $\mathbf{x}^*$ 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 $\{\Theta_n\}$ appropriately. If it grows too quickly, the results might be very similar to that of the original SA. 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 $\mathbf{x}_1 \in \Theta_1$, $\{a_n\}$ and $\{\Theta_n\}$.
- Initialize. Let $n = 1$ and $m = 1$.
- While $n < N$,
  - Step 1. Generate an estimate $\widehat{\nabla} J(\mathbf{x}_n)$ of $\nabla J(\mathbf{x}_n)$.
  - Step 2. Compute $\mathbf{x}'_{n+1} = \mathbf{x}_n - a_n \widehat{\nabla} J(\mathbf{x}_n)$. If $\mathbf{x}'_{n+1} \in \Theta_m$, go to Step 3. Otherwise, go to Step 4.
  - Step 3. Let $\mathbf{x}_{n+1} = \mathbf{x}'_{n+1}$, $n = n + 1$ and go to Step 1.
  - Step 4. Let $\mathbf{x}_{n+1} = \Pi_{\Theta_m}(\mathbf{x}'_{n+1})$, $n = n + 1,\ m = m + 1$ and go to Step 1.
- Output. $\mathbf{x}^*_N = \mathbf{x}_N$.

Theorem 2.2.6. *(Theorem 2 [2]) Let the sequence $\{\mathbf{x}_n\}$ be generated using the above algorithm, $\epsilon_n = \widehat{\nabla} J(\mathbf{x}) - E[\widehat{\nabla} J(\mathbf{x})|\mathcal{F}_n]$, and $\beta_n = E[\widehat{\nabla} J(\mathbf{x})|\mathcal{F}_n] - \nabla J(\mathbf{x})$, where $\mathcal{F}_n$ is the smallest σ-algebra used to generate $\mathbf{x}_{n+1}$. If the following conditions hold:*

1. *The sequence $\{\Theta_k\}$ is a set of compact convex sets such that $\Theta_k \subseteq \Theta_{k+1}$ for all k and $\cup_{k=1}^{\infty} \Theta_k = \Theta$.*
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* $E[||\epsilon_n||^2|\mathcal{F}_n] \leq \frac{\kappa}{c_n^2}(1 + ||\mathbf{x}_n - \mathbf{x}^*||^2)$ *a.s. for all* n.

4. $||\beta_n||$ *is bounded a.s. and* $\sum_{n=1}^{\infty} a_n||\beta_n|| < \infty$ *a.s.*

5. *There exist a positive sequence of real numbers* $\{M_n\}$ *an integer* $N \geq 1$ *such that* $\sum_{n=1}^{\infty} a_n^2 M_n^2 < \infty$ *and for all* $n \geq N$, $\sup_{\mathbf{x}\in\Theta_{n-1}} ||J(\mathbf{x})|| \leq M_n$.

6. *There exists a unique* $\mathbf{x}^* \in \Theta$ *such that* $\nabla J(\mathbf{x}^*) = \mathbf{0}$, *and for all* $0 < \delta \leq 1$, $\inf_{\mathbf{x}\in\Theta:\delta\leq||\mathbf{x}-\mathbf{x}^*||\leq\delta^{-1}} J(\mathbf{x})^T(\mathbf{x} - \mathbf{x}^*) > 0$.

Then $\mathbf{x}_n \to \mathbf{x}^*$ *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.

Chapter 3

New Hybrid Stochastic Approximation Methods

3.1 Motivation

Theoretically, unbiased direct gradients lead to faster asymptotic convergence rates in SA algorithms compared to indirect gradient estimates, but this does not necessarily translate to better finite-time performance. In the deterministic setting, where the true gradient is known, indirect gradients are not useful, whereas in the stochastic setting, direct gradients are noisy and indirect gradients actually contain information that can be used to better approximate the gradient and in turn accelerate the SA convergence to a neighborhood of the optimum. To the best of our knowledge, the current SA algorithms either use direct gradients or indirect gradients but not both simultaneously. Our algorithms exploit the additional information provided by indirect gradients by using a gradient that integrates both direct and indirect gradient estimates.

In this chapter, we do the following:

1. We propose two new hybrid stochastic approximation algorithms, STAR-SA and STAR-SPSA, which incorporate both direct and indirect gradient estimates, and are provably convergent.

2. We derive variance minimizing weights for the STAR-SA and STAR-SPSA

gradient estimates.

3. We show that the variance of the STAR-SA gradient is lower than that of RM and KW, and that the variance of the STAR-SPSA gradient is less than that of RM and SPSA, for a deterministic weight sequence.

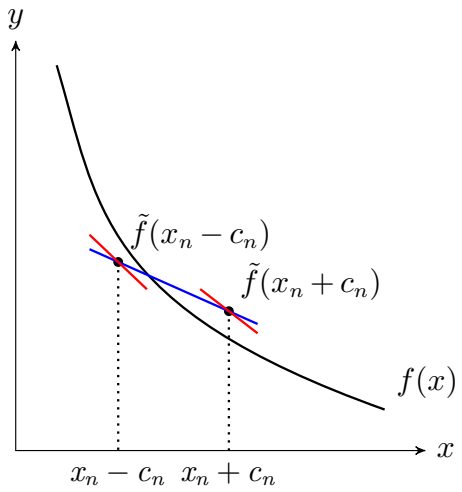
4. For simple quadratic functions, we show that the MSE of the estimate generated using STAR-SA is strictly less than that of RM and KW under certain conditions.

5. We illustrate the robustness/effectiveness of STAR-SA and STAR-SPSA through numerical experiments.

3.2 Secant-Tangents AveRaged Stochastic Approximation

In this section, we introduce the one-dimensional **S**ecant-**T**angents **A**ve**R**aged stochastic approximation (STAR-SA) algorithm, which incorporates both direct and indirect gradient estimates [**?**, 10]. Figure 3.1 illustrates the STAR gradient estimator, which uses estimates of the sample performance $\tilde{f}$ and its gradient $\tilde{f}'$ at two points, $x_n + c_n$ and $x_n - c_n$, where the values $\tilde{f}(x_n + c_n)$ and $\tilde{f}(x_n - c_n)$ do not lie on the graph of the true function due to estimation noise error. For the indirect gradient, the values $\tilde{f}(x_n+c_n)$ and $\tilde{f}(x_n-c_n)$ are used to compute a symmetric finite difference (**S**ecant). The two direct gradient estimates, $\tilde{f}'(x_n + c_n)$ and $\tilde{f}'(x_n - c_n)$, can be averaged to provide another gradient estimate (**T**angents **A**ve**R**aged).

The **S**ecant-**T**angents **A**ve**R**aged stochastic approximation (STAR-SA) gra-



Secant

$$g_S \equiv \frac{\tilde{f}(x_n + c_n) - \tilde{f}(x_n - c_n)}{2c_n}$$

Tangents **A**ve**R**aged

$$g_{TAR} \equiv \frac{\tilde{f}'(x_n + c_n) + \tilde{f}'(x_n - c_n)}{2}$$

Figure 3.1: Illustration of STAR gradient, where $\tilde{f}$ and $\tilde{f}'$ are estimates of f and f', respectively.

dient estimator is a convex combination of a symmetric finite difference gradient estimator (secant) and an average of two direct gradient estimators (tangents):

$$g_{STAR}(x_n) = \alpha_n \frac{\tilde{f}(x_n + c_n) - \tilde{f}(x_n - c_n)}{2c_n} + (1 - \alpha_n)\frac{\tilde{f}'(x_n + c_n) + \tilde{f}'(x_n - c_n)}{2}, \quad (3.1)$$

where $\alpha_n \in [0, 1]$ and $c_n \in \mathbb{R}^+$ for all n, and $c_n \to 0$ as $n \to \infty$.

The weight α_n in the convex combination is crucial in the performance of STAR-SA. In fact, for any non-zero and finite variance level of the function and its gradient, there exists a sequence $\{\alpha_n\}$ such that the variance of the STAR-SA gradient estimator is less than the individual gradient estimates, **S**ecant g_S and **T**angents **A**ve**R**aged g_{TAR}.

Now we present theoretical results pertaining to STAR-SA. Let $\tilde{f}(x_n \pm c_n) = f(x_n \pm c_n) + \epsilon_n^{\pm}$ and $\tilde{f}'(x_n \pm c_n) = f'(x_n \pm c_n) + \delta_n^{\pm}$, where $\epsilon_n^{\pm}$ and $\delta_n^{\pm}$ are Gaussian noise

with zero mean. For notational simplicity, let $Var(\tilde{f}(x_n)) = \sigma_f^2$, $Var(\tilde{f}'(x_n)) = \sigma_g^2$, $Var(g_{STAR}(x_n)) = \sigma_n^2$, $Var(g_S(x_n)) = Var\left(\frac{\tilde{f}(x_n+c_n)-\tilde{f}(x_n-c_n)}{2c_n}\right) = \frac{\sigma_f^2}{2c_n^2}$, $Var(g_{TAR}(x_n))$ $= Var\left(\frac{\tilde{f}'(x_n+c_n)+\tilde{f}'(x_n-c_n)}{2}\right) = \frac{\sigma_g^2}{2}$, and $Corr(\tilde{f}(x_n), \tilde{f}'(x_n)) = \rho$.

3.2.1 Optimal Convex Weight

3.2.1.1 Homogeneous Noise

The following provides a particular weight for minimizing the variance of the STAR-SA gradient.

Lemma 3.2.1. *For g_{STAR} defined in (3.1), assume for all n, $\epsilon_n^+ \perp \epsilon_n^-$ and $\delta_n^+ \perp \delta_n^-$. Then $Var(g_{STAR}(x_n))$ is minimized when*

$$\alpha_n^* = \frac{\sigma_g^2 c_n^2}{\sigma_f^2 + \sigma_g^2 c_n^2}. \tag{3.2}$$

Proof. By the orthogonality assumption, the variance of the STAR-SA gradient (3.1) can be written as

$$\begin{aligned}
\sigma_n^2 &= Var\left(\alpha_n \frac{\tilde{f}(x_n+c_n)-\tilde{f}(x_n-c_n)}{2c_n} + (1-\alpha_n)\frac{\tilde{f}'(x_n+c_n)+\tilde{f}'(x_n-c_n)}{2}\right) \\
&= Var\left(\frac{\alpha_n}{2c_n}\tilde{f}(x_n+c_n) + \frac{1-\alpha_n}{2}\tilde{f}'(x_n+c_n)\right) \\
&\qquad + Var\left(-\frac{\alpha_n}{2c_n}\tilde{f}(x_n-c_n) + \frac{1-\alpha_n}{2}\tilde{f}'(x_n-c_n)\right) \\
&= \left(\frac{\alpha_n}{2c_n}\right)^2\sigma_f^2 + \left(\frac{1-\alpha_n}{2}\right)^2\sigma_g^2 + 2\frac{\alpha_n}{2c_n}\frac{1-\alpha_n}{2}\rho\sigma_f\sigma_g \\
&\qquad + \left(\frac{\alpha_n}{2c_n}\right)^2\sigma_f^2 + \left(\frac{1-\alpha_n}{2}\right)^2\sigma_g^2 - 2\frac{\alpha_n}{2c_n}\frac{1-\alpha_n}{2}\rho\sigma_f\sigma_g \\
&= \frac{\alpha_n^2}{2c_n^2}\sigma_f^2 + \frac{(1-\alpha_n)^2}{2}\sigma_g^2,
\end{aligned} \tag{3.3}$$

where the second equality follows from the independence of the function/gradient at different points. Differentiating (3.3) with respect to α_n, we find

$$\frac{\partial \sigma_n^2}{\partial \alpha_n} = \frac{\alpha_n}{c_n^2}\sigma_f^2 - (1-\alpha_n)\sigma_g^2. \tag{3.4}$$

Setting (3.4) equal to zero and solving for α_n yields (3.2). Since (3.4) is monotonically increasing in α_n, (3.2) is the global minimum. □

The following result shows that the variance of the STAR-SA gradient is provably less than the variance of the gradients generated under RM and KW.

Proposition 3.2.2. *Let the conditions of Lemma 3.2.1 hold. If the convex weight in (3.1) is given by (3.2), then* $Var(g_{STAR}(x_n)) \leq \min\{Var(g_S(x_n)), Var(g_{TAR}(x_n))\}$ *for all n, where the inequality is strict if* $\sigma_f, \sigma_g > 0$.

Proof. Under the conditions of Lemma 3.2.1, the variance of the STAR-SA gradient (3.1) can be written as

$$\sigma_n^2 \;=\; \frac{\alpha_n^2}{2c_n^2}\sigma_f^2 + \frac{(1-\alpha_n)^2}{2}\sigma_g^2. \tag{3.5}$$

Substitute α_n^* from (3.2) into (3.5) for α_n to obtain

$$\sigma_n^2 = \frac{1}{2c_n^2}\left(\frac{\sigma_g^2 c_n^2}{\sigma_f^2+\sigma_g^2 c_n^2}\right)^2 \sigma_f^2 + \frac{1}{2}\left(1-\frac{\sigma_g^2 c_n^2}{\sigma_f^2+\sigma_g^2 c_n^2}\right)^2 \sigma_g^2 = \frac{\sigma_f^2\sigma_g^2}{2\left(\sigma_f^2+\sigma_g^2 c_n^2\right)}.$$

To show σ_n^2 is strictly less than $\min\{Var(g_S(x_n)), Var(g_{TAR}(x_n))\}$, consider two cases: 1) $\sigma_n^2 < \frac{\sigma_f^2}{2c_n^2}$ and 2) $\sigma_n^2 < \frac{\sigma_g^2}{2}$. However, since $c_n \to 0$ as $n \to \infty$, then $\frac{\sigma_f^2}{2c_n^2} \to \infty$, so eventually we will be in the latter case.

Case 1: If $\sigma_n^2 < \frac{\sigma_f^2}{2c_n^2}$, then $\frac{\sigma_f^2\sigma_g^2}{2\left(\sigma_f^2+\sigma_g^2 c_n^2\right)} < \frac{\sigma_f^2}{2c_n^2}$, which holds when $0 < \sigma_f^2$.

Case 2: If $\sigma_n^2 < \frac{\sigma_g^2}{2}$, then $\frac{\sigma_f^2\sigma_g^2}{2\left(\sigma_f^2+\sigma_g^2 c_n^2\right)} < \frac{\sigma_g^2}{2}$, which holds when $0 < \sigma_g^2 c_n^2$.

If $\sigma_f, \sigma_g > 0$, then the inequality is strict; however, if $\sigma_f \sigma_g = 0$, then the same arguments hold with " $<$ " replaced with " $\leq$ " for the non-strict case. Therefore, $\sigma_n^2 \leq \min\left\{\frac{\sigma_f^2}{2c_n^2}, \frac{\sigma_g^2}{2}\right\} = \min\{Var(g_S(x_n)), Var(g_{TAR}(x_n))\}$, and the inequality is strict for $\sigma_f, \sigma_g > 0$. $\square$

The weight α_n^* depends on the function and gradient variances, and the finite difference perturbation size. These values, with the exception of the perturbation size, are generally unknown and must be estimated. Although the orthogonality condition in Lemma 3.2.1 allows the function and its gradient to be correlated, the optimal weight does not depend on the correlation coefficient, due to the homogeneous noise assumption. If we eliminate this assumption to allow for non-homogeneous noise, the weight will depend on the correlation. Also, the STAR-SA gradient estimate converges to the average of the direct gradient estimates, since $\alpha_n^* \to 0$ as $n \to 0$.

3.2.1.2 Non-homogeneous Noise

For non-homogeneous noise, σ_f and σ_g depend on x, and we write $Var(\tilde{f}_n^{\pm}) = \sigma_{f,n^{\pm}}^2$, $Var(\tilde{g}_n^{\pm}) = \sigma_{g,n^{\pm}}^2$, $Corr(\tilde{f}_n^-, \tilde{g}_n^+) = Corr(\tilde{f}_n^+, \tilde{g}_n^-) = 0$, $Corr(\tilde{f}_n^+, \tilde{g}_n^+) = \rho_n^+$ and $Corr(\tilde{f}_n^-, \tilde{g}_n^-) = \rho_n^-$, with $\tilde{f}_n^{\pm} = \tilde{f}(x_n \pm c_n) = f(x_n \pm c_n) + \epsilon_n^{\pm}$ and $\tilde{g}_n^{\pm} = \tilde{f}'(x_n \pm c_n) = f'(x_n \pm c_n) + \delta_n^{\pm}$, where $\epsilon^{\pm}$ and $\delta_n^{\pm}$ is Gaussian noise with zero mean.

Lemma 3.2.3. *If the conditions of Lemma 3.2.1 and* $\rho_n^- \sigma_{f,n^-} \sigma_{g,n^-} - \rho_n^+ \sigma_{f,n^+} \sigma_{g,n^+} \geq 0$ *hold, then* $Var(g_{STAR}(x_n))$ *is minimized when*

$$\alpha_n^* = \frac{c_n^2(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) + 2c_n(\rho_n^- \sigma_{f,n^-} \sigma_{g,n^-} - \rho_n^+ \sigma_{f,n^+} \sigma_{g,n^+})}{\sigma_{f,n^+}^2 + \sigma_{f,n^-}^2 + c_n^2(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) + 2c_n(\rho_n^- \sigma_{f,n^-} \sigma_{g,n^-} - \rho_n^+ \sigma_{f,n^+} \sigma_{g,n^+})}. \quad (3.6)$$

Proof. The variance of the STAR-SA gradient can be written as

$$\begin{aligned}
\sigma_n^2 &= Var\left(\alpha_n \frac{\tilde{f}_n^+ - \tilde{f}_n^-}{2c_n} + (1-\alpha_n)\frac{\tilde{g}_n^+ + \tilde{g}_n^-}{2}\right) \\
&= Var\left(\frac{\alpha_n}{2c_n}\tilde{f}_n^+ + \frac{1-\alpha}{2}\tilde{g}_n^+\right) + Var\left(-\frac{\alpha_n}{2c_n}\tilde{f}_n^- + \frac{1-\alpha}{2}\tilde{g}_n^-\right) \\
&= \left(\frac{\alpha_n}{2c_n}\right)^2 Var(\tilde{f}_n^+) + \left(\frac{1-\alpha_n}{2}\right)^2 Var(\tilde{g}_n^+) + 2\frac{\alpha_n}{2c_n}\frac{1-\alpha_n}{2}Cov(\tilde{f}_n^+, \tilde{g}_n^+) \\
&\quad + \left(\frac{\alpha_n}{2c_n}\right)^2 Var(\tilde{f}_n^-) + \left(\frac{1-\alpha_n}{2}\right)^2 Var(\tilde{g}_n^-) - 2\frac{\alpha_n}{2c_n}\frac{1-\alpha_n}{2}Cov(\tilde{f}_n^-, \tilde{g}_n^-) \\
&= \frac{\alpha_n^2}{4c_n^2}\left(\sigma_{f,n^+}^2 + \sigma_{f,n^-}^2\right) + \frac{(1-\alpha_n)^2}{4}(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) \\
&\qquad + \frac{\alpha_n(1-\alpha_n)}{2c_n}\left(\rho_n^+\sigma_{f,n^+}\sigma_{g,n^+} - \rho_n^-\sigma_{f,n^-}\sigma_{g,n^-}\right), \qquad (3.7)
\end{aligned}$$

where the second equality follows from the independence of the function/gradient at different points. Differentiate (3.7) with respect to the convex weight α_n to obtain

$$\begin{aligned}
\frac{\partial V}{\partial \alpha_n} &= \frac{\alpha_n}{2c_n^2}\left(\sigma_{f,n^+}^2 + \sigma_{f,n^-}^2\right) - \frac{(1-\alpha_n)}{2}(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) \\
&\qquad + \left(\frac{1-2\alpha_n}{2c_n}\right)(\rho_n^+\sigma_{f,n^+}\sigma_{g,n^+} - \rho_n^-\sigma_{f,n^-}\sigma_{g,n^-}),
\end{aligned}$$

which is monotonically increasing in α_n. Set $\frac{\partial V}{\partial \alpha_n} = 0$ to obtain

$$\begin{aligned}
&\alpha_n\left(\frac{\sigma_{f,n^+}^2 + \sigma_{f,n^-}^2}{2c_n^2} + \frac{\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2}{2} + \frac{\rho_n^-\sigma_{f,n^-}\sigma_{g,n^-} - \rho_n^+\sigma_{f,n^+}\sigma_{g,n^+}}{c_n}\right) \\
&\qquad = \frac{\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2}{2} + \frac{\rho_n^-\sigma_{f,n^-}\sigma_{g,n^-} - \rho_n^+\sigma_{f,n^+}\sigma_{g,n^+}}{c_n},
\end{aligned}$$

so the optimal weight is

$$\begin{aligned}
\alpha_n^* &= \frac{2c_n^2}{\sigma_{f,n^+}^2 + \sigma_{f,n^-}^2 + c_n^2(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) + 2c_n(\rho_n^-\sigma_{f,n^-}\sigma_{g,n^-} - \rho_n^+\sigma_{f,n^+}\sigma_{g,n^+})} \cdot \\
&\qquad \frac{c_n(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) + 2\rho_n^-\sigma_{f,n^-}\sigma_{g,n^-} - 2\rho_n^+\sigma_{f,n^+}\sigma_{g,n^+}}{2c_n} \\
&= \frac{c_n^2(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) + 2c_n(\rho_n^-\sigma_{f,n^-}\sigma_{g,n^-} - \rho_n^+\sigma_{f,n^+}\sigma_{g,n^+})}{\sigma_{f,n^+}^2 + \sigma_{f,n^-}^2 + c_n^2(\sigma_{g,n^+}^2 + \sigma_{g,n^-}^2) + 2c_n(\rho_n^-\sigma_{f,n^-}\sigma_{g,n^-} - \rho_n^+\sigma_{f,n^+}\sigma_{g,n^+})}. \square
\end{aligned}$$

Now we prove Proposition 3.2.2 for the non-homogeneous setting. For notation simplicity, $F_n = \sigma^2_{f,n^+} + \sigma^2_{f,n^-}$, $G_n = \sigma^2_{g,n^+} + \sigma^2_{g,n^-}$, $P_n = \rho^-_n \sigma_{f,n^-}\sigma_{g,n^-} - \rho^+_n \sigma_{f,n^+}\sigma_{g,n^+}$, and $D_n = \sigma^2_{f,n^+} + \sigma^2_{f,n^-} + c_n^2(\sigma^2_{g,n^+} + \sigma^2_{g,n^-}) + 2c_n(\rho^-_n \sigma_{f,n^-}\sigma_{g,n^-} - \rho^+_n \sigma_{f,n^+}\sigma_{g,n^+})$.

Let the conditions of Lemma 3.2.1 hold, then the variance of the STAR-SA gradient (3.1) can be written as

$$\begin{aligned}
\sigma_n^2 &= \frac{\alpha_n^2}{4c_n^2}\left(\sigma^2_{f,n^+} + \sigma^2_{f,n^-}\right) + \frac{(1-\alpha_n)^2}{4}(\sigma^2_{g,n^+} + \sigma^2_{g,n^-}) \\
&\qquad + \frac{\alpha_n(1-\alpha_n)}{2c_n}\left(\rho^+_n \sigma_{f,n^+}\sigma_{g,n^+} - \rho^-_n \sigma_{f,n^-}\sigma_{g,n^-}\right) \\
&= \frac{\alpha_n^2}{4c_n^2}F_n + \frac{(1-\alpha_n)^2}{4}G_n - \frac{\alpha_n(1-\alpha_n)}{2c_n}P_n. \qquad (3.8)
\end{aligned}$$

Substitute the weight α_n^* from (3.6) into each of the three components in (3.8) to obtain

$$\begin{aligned}
\sigma_n^2 &= \frac{c_n^4 G_n^2 + 4c_n^3 G_n P_n + 4c_n^2 P_n^2}{4c_n^2 D_n^2}F_n + \frac{F_n^2 G_n}{4D_n^2} - \frac{c_n^2 G_n F_n + 2c_n P_n F_n}{2c_n D_n^2}P_n \\
&= \frac{c_n^2 G_n^2 F_n + 4c_n G_n P_n F_n + 4P_n^2 F_n}{4D_n^2} + \frac{F_n^2 G_n}{4D_n^2} - \frac{2c_n G_n F_n P_n + 4P_n^2 F_n}{4D_n^2} \\
&= \frac{F_n^2 G_n + 2c_n G_n P_n F_n + c_n^2 G_n^2 F_n}{4D_n^2}. \qquad (3.9)
\end{aligned}$$

To show σ_n^2 is strictly less than $\min\{Var(g_S(x_n)), Var(g_{TAR}(x_n))\}$, consider two cases:

Case 1: If $Var(g_{STAR}(x_n)) < Var(g_S(x_n))$, then $\frac{F_n^2 G_n + 2c_n G_n P_n F_n + c_n^2 G_n^2 F}{4D_n^2} < \frac{F_n}{4c_n^2}$, which can be simplified to $0 < F_n^3 + 4c_n F_n^2 P_n + c_n^2(4P_n^2 F_n + F_n^2 G_n) + 4c_n^3 G_n P_n F_n$, and is satisfied if $F_n > 0$ and $P_n \geq 0$.

Case 2: If $Var(g_{STAR}(x_n)) < Var(g_{TAR}(x_n)$, then $\frac{F_n^2 G_n + 2c_n G_n P_n F_n + c_n^2 G_n^2 F_n}{4D_n^2} < \frac{G_n}{4}$, which simplifies to $0 < c_n^4 G_n^3 + 4c_n^3 P_n G_n^2 + c_n^2(4P_n^2 G_n + F_n G_n^2) + 2c_n F_n P_n G_n$, and holds for $G_n > 0$ and $P_n \geq 0$.

The same argument holds for the non-strict case by replacing " $<$ " with " $\leq$ " in both cases and are both satisfied for $P_n \geq 0$. Therefore, $\sigma_n^2 \leq \min\{Var(g_S(x_n)), Var(g_{TAR}(x_n))\}$, where the inequality is strict for $F_n, G_n > 0$. $\square$

3.2.2 Convergence

The next result establishes mean-squared convergence of the STAR-SA algorithm. The proof closely follows [18] and [62].

Theorem 3.2.4. *Let $\{x_n\}$ be a sequence following recursion (1.2) with $\Theta = \mathbb{R}$ and $\widehat{\nabla} f(x_n)$ as the STAR gradient estimate defined in (3.1). If the following conditions hold:*

1. *There exist positive sequences $\{a_n\}$ and $\{c_n\}$ such that $\sum_{n=1}^{\infty} a_n = \infty$, $\sum_{n=1}^{\infty} a_n c_n < \infty$, $\sum_{n=1}^{\infty} a_n^2 < \infty$, and $\sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty$.*

2. *There exist $K_0, K_1 > 0$ such that $K_0|x - x^*| \leq |f'(x)| \leq K_1|x - x^*|$ for all $x \in \Theta$.*

3. *$f'(x)(x - x^*) > 0$ for all $x \in \Theta \backslash \{x^*\}$.*

4. *For $c > 0$, $\sigma^2 = \sup_{x \in \Theta} Var[\tilde{f}(x + c) - \tilde{f}(x - c)|x] < \infty$ for all $x \in \Theta$.*

5. *ϵ_n^+, ϵ_n^-, δ_n^+, δ_n^- are i.i.d. with mean zero for all n.*

Then x_n converges to x^* in L^2 as $n \to \infty$.

Proof. Fix positive sequences $\{a_n\}$ and $\{c_n\}$ satisfying condition 1. Let $\{x_n\}$ be a sequence of estimates following recursion (1.2) with $\Theta = \mathbb{R}$ and $\widehat{\nabla} f(x_n) = g_{STAR}(x_n)$. As defined in Section 3.2, $g_{STAR} = \alpha g_S + (1 - \alpha) g_{TAR}$.

The mean-squared error of the nth estimate can be expressed as

$$\begin{aligned} E\left[(x_{n+1}-x^*)^2\right] &= E[(x_n - x^* - a_n g_{STAR}(x_n))^2] \\ &= E[(x_n-x^*)^2] - 2a_n E[(x_n-x^*)g_{STAR}(x_n)] + a_n^2 E[g_{STAR}^2(x_n)]. \end{aligned} \tag{3.10}$$

Now we establish necessary bounds in order to generate an upper bound for (3.10). By the mean-value theorem, there exist ξ_1 and ξ_2 such that $0 \le \xi_1, \xi_2 \le 1$ where $f(x_n+c_n) = f(x_n)+f'(x_n+\xi_1 c_n)c_n$ and $f(x_n-c_n) = f(x_n)-f'(x_n-\xi_2 c_n)c_n$.

$$\begin{aligned} E[g_S(x_n)|x_n] &= \frac{f(x_n+c_n)-f(x_n-c_n)}{2c_n} \\ &= \frac{f'(x_n+\xi_1 c_n)+f'(x_n-\xi_2 c_n)}{2} \qquad (3.11) \\ &= (x_n-x^*)\frac{1}{2}\left[\frac{f'(x_n+\xi_1 c_n)}{x_n-x^*+\xi_1 c_n}+\frac{f'(x_n-\xi_2 c_n)}{x_n-x^*-\xi_2 c_n}\right] \\ &\quad + \frac{c_n}{2}\left[\xi_1\frac{f'(x_n+\xi_1 c_n)}{x_n-x^*+\xi_1 c_n} - \xi_2\frac{f'(x_n-\xi_2 c_n)}{x_n-x^*-\xi_2 c_n}\right] \\ &\le |x_n-x^*|K_1 + c_n K_1, \qquad (3.12) \end{aligned}$$

where the last inequality follows from

$$K_0 \le \left|\frac{f'(x)}{x-x^*}\right| = \frac{f'(x)}{x-x^*} \le K_1, \tag{3.13}$$

as a result of combining conditions 2 and 3. By applying (3.13),

$$\begin{aligned} E[(x_n-x^*)g_S(x_n)|x_n] &= (x_n-x^*)^2\frac{1}{2}\left[\frac{f'(x_n+\xi_1 c_n)}{x_n-x^*+\xi_1 c_n}+\frac{f'(x_n-\xi_2 c_n)}{x_n-x^*-\xi_2 c_n}\right] \\ &\quad + \frac{c_n}{2}(x_n-x^*)\left[\xi_1\frac{f'(x_n+\xi_1 c_n)}{x_n-x^*+\xi_1 c_n} - \xi_2\frac{f'(x_n-\xi_2 c_n)}{x_n-x^*-\xi_2 c_n}\right] \\ &\ge (x_n-x^*)^2 K_0 - |x_n-x^*|c_n K_1. \qquad (3.14) \end{aligned}$$

Using conditional variance, condition 4, and after applying $|a+b|^r \le 2^{r-1}(|a|^r+|b|^r)$

for $r > 1$ to the square of equation (3.12) yields

$$\begin{aligned} E[g_S(x_n)^2|x_n] &= Var(g_S(x_n)|x_n) + E[g_S(x_n)|x_n]^2 \\ &\leq \frac{\sigma^2}{4c_n^2} + 2|x_n - x^*|^2 K_1^2 + 2c_n^2 K_1^2. \end{aligned} \tag{3.15}$$

The bounds regarding $g_{TAR}(x_n)$ are identical to those of $g_S(x_n)$, since $g_{TAR}(x_n)$ is equal to the RHS of (3.11) with $\xi_1 = \xi_2 = 1$. Using (3.12), (3.14), and (3.15) yields

$$\begin{aligned} E\left[(x_{n+1} - x^*)^2|x_n\right] &= E[(x_n - x^* - a_n g_{STAR}(x_n))^2|x_n] \\ &= E[(x_n - x^*)^2|x_n] - 2a_n E[(x_n - x^*)g_{STAR}(x_n)|x_n] + a_n^2 E[g_{STAR}^2(x_n)|x_n] \\ &\leq (x_n - x^*)^2 - 2a_n\left((x_n - x^*)^2 K_0 - |x_n - x^*|c_n K_1\right) \\ &\qquad + 4a_n^2\left(\frac{\sigma^2}{4c_n^2} + 2|x_n - x^*|^2 K_1^2 + 2c_n^2 K_1^2\right) \\ &\leq (x_n - x^*)^2\left(1 - 2a_n K_0 + 8a_n^2 K_1^2\right) + 2a_n c_n K_1 |x_n - x^*| + \frac{a_n^2\sigma^2}{c_n^2} \\ &\qquad + 8a_n^2 c_n^2 K_1^2. \end{aligned} \tag{3.16}$$

Define $b_n = E[(x_n - x^*)^2]$. Take the expectation of (3.16) and apply the inequalities $E[\sqrt{b_n}] \leq \sqrt{E[b_n]}$ and $\sqrt{b_n} \leq b_n + 1$ to obtain

$$\begin{aligned} E[(x_{n+1} - x^*)^2] &\leq E[(x_n - x^*)^2]\left(1 - 2a_n K_0 + 8a_n^2 K_1^2\right) + 2a_n c_n K_1 E[|x_n - x^*|] \\ &\qquad + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2 c_n^2 K_1^2 \\ &\leq b_n\left(1 - 2a_n K_0 + 8a_n^2 K_1^2\right) + 2a_n c_n K_1\sqrt{b_n} + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2 c_n^2 K_1^2 \\ &\leq b_n\left(1 - 2a_n K_0 + 8a_n^2 K_1^2\right) + 2a_n c_n K_1(b_n + 1) + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2 c_n^2 K_1^2 \\ &= b_n\left(1 - 2a_n K_0 + 2a_n c_n K_1 + 8a_n^2 K_1^2\right) + 2a_n c_n K_1 + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2 c_n^2 K_1^2 \\ &= b_n R_n + T_n, \end{aligned}$$

where $R_n = 1 - 2a_n K_0 + 2a_n c_n K_1 + 8a_n^2 K_1^2 \geq 1 - 2a_n K_0 + 8a_n^2 K_1^2 > 0$, which

is quadratic in a_n, and the discriminant $4K_0^2 - 16K_1^2$ is negative if $K_0 < 2\sqrt{2}K_1$, which holds by condition 2. The coefficient for a_n in R_n, $2c_nK_1 - 2K_0$, is negative for sufficiently large n since $c_n \to 0$ as $n \to \infty$. By condition 1, $\prod_{n=1}^{\infty} R_n < \infty$ and $\sum_{n=1}^{\infty} T_n < \infty$. As a result, all partial products are uniformly bounded. Thus, $b_n \leq M < \infty$. Since $R_i > 0$ for all i, we have the recursion

$$b_{n+1} \leq b_1 \prod_{i=1}^{n} R_i + \sum_{i=2}^{n-1} \left(T_i \prod_{j=i+1}^{n} R_j \right) + T_n.$$

Applying $b_n \leq M$ to the second terms in both (3.14) and (3.15) yields

$$\begin{aligned}
&E[(x_{n+1} - x^*)^2] \\
&\leq E[(x_n - x^*)^2] \left(1 - 2a_nK_0 + 8a_n^2K_1^2\right) + 2a_nc_nK_1E[|x_n - x^*|] + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2c_n^2K_1^2 \\
&= E[(x_n - x^*)^2] \left(1 - 2a_nK_0\right) + 8a_n^2K_1^2E[(x_n - x^*)^2] + 2a_nc_nK_1E[|x_n - x^*|] \\
&\qquad + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2c_n^2K_1^2 \\
&\leq E[(x_n - x^*)^2] \left(1 - 2a_nK_0\right) + 8a_n^2K_1^2M + 2a_nc_nK_1\sqrt{M} + \frac{a_n^2\sigma^2}{c_n^2} + 8a_n^2c_n^2K_1^2 \\
&= b_nR_n' + T_n'.
\end{aligned}$$

Since $a_n \to 0$, there exists n_0 such that $a_n < 1/(2K_0)$ for all $n \geq n_0$, which implies $R_n' > 0$. Then we have the recursion

$$b_{n+1} \leq b_{n_0} \prod_{i=n_0}^{n} R_i' + \sum_{i=n_0}^{n-1} \left(T_i' \prod_{j=i+1}^{n} R_j' \right) + T_n' \text{ for } n = n_0 + 1, n_0 + 2, ...$$

$\prod_{i=n_0}^{n} R_i' \to 0$ as $n \to \infty$ since $\sum a_n = \infty$. By condition 1 and Kronecker's lemma, $\sum_{i=n_0}^{n-1} \left(T_i' \prod_{j=i+1}^{n} R_j' \right) \to 0$ as $n \to \infty$. Therefore, for $\epsilon > 0$, there exists n_1 such that $\prod_{j=n_0}^{n} R_j' < \frac{\epsilon}{3M^2}$ for all $n \geq n_1$, there exists n_2 such that $\sum_{i=n_0}^{n-1} \left(T_i' \prod_{j=i+1}^{n} R_j' \right) < \epsilon/3$ for all $n \geq n_2$, and there exists n_3 such that $T_n' < \epsilon/3$ for all $n \geq n_3$. Hence, for all $n > \max(n_1, n_2, n_3) > n_0$, $b_{n+1} \leq M^2 \frac{\epsilon}{3M^2} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon$. $\square$

We investigate the exact MSE for quadratic functions of the form $f(x) = -ax^2$, where $a > 0$ and $x^* = 0$. The finite-time MSE of quadratic functions in this form can be computed explicitly, which allows us to compare the performance of different algorithms analytically.

Proposition 3.2.5. *For $f(x) = -ax^2$, $a > 0$, the respective MSEs after n iterations following recursion (1.2) can be expressed as*

$$\begin{aligned}
MSE_{RM} &= A_n + \sigma_g^2 \sum_{k=1}^{n-2} a_k^2 \Pi_{j=k+1}^{n-1} (1 - 2a_j a)^2 + \sigma_g^2 a_{n-1}^2, \\
MSE_{KW} &= A_n + \frac{\sigma_f^2}{2} \sum_{k=1}^{n-2} \left(\frac{a_k^2}{c_k^2} \right) \Pi_{j=k+1}^{n-1} (1 - 2a_j a)^2 + \frac{\sigma_f^2 a_{n-1}^2}{2c_{n-1}^2}, \\
MSE_{STAR} &= A_n + \sum_{k=1}^{n-2} \frac{a_k^2}{2} \left(\frac{\alpha_k^2 \sigma_f^2}{c_k^2} + (1 - \alpha_k)^2 \sigma_g^2 \right) \cdot \Pi_{j=k+1}^{n-1} (1 - 2a_j a)^2 \\
&\qquad + \frac{a_{n-1}^2}{2} \left(\frac{\alpha_{n-1}^2 \sigma_f^2}{c_{n-1}^2} + (1 - \alpha_{n-1})^2 \sigma_g^2 \right),
\end{aligned}$$

where $A_n = x_0^2 \Pi_{i=1}^{n-1} (1 - 2a_i a)^2$ and x_0 is the initial iterate.

Proof. We can express the nth estimates of RM, KW, and STAR-SA respectively as

$$\begin{aligned}
x_n^{RM} &= x_{n-1} + a_{n-1} \tilde{f}(x_{n-1}) \\
&= x_{n-1} + a_{n-1}(-2ax_{n-1} + \delta_{n-1}) \\
&= x_{n-1}(1 - 2a_{n-1}a) + a_{n-1}\delta_{n-1} \\
x_n^{KW} &= x_{n-1} + a_{n-1} \tilde{f}(x_{n-1}) \\
&= x_{n-1} + a_{n-1} \left(-2ax_{n-1} + \frac{\epsilon_{n-1}^+ - \epsilon_{n-1}^-}{2c_{n-1}} \right) \\
&= x_{n-1}(1 - 2a_{n-1}ax_{n-1}) + a_{n-1} \frac{\epsilon_{n-1}^+ - \epsilon_{n-1}^-}{2c_{n-1}}
\end{aligned}$$

$$
\begin{aligned}
x_n^{STAR} &= x_{n-1} + a_{n-1}\tilde{f}(x_{n-1}) \\
&= x_{n-1} + a_{n-1}\left(-2ax_{n-1} + \alpha_{n-1}\frac{\epsilon_{n-1}^{+} - \epsilon_{n-1}^{-}}{2c_{n-1}} + (1-\alpha_{n-1})\delta_{n-1}\right) \\
&= x_{n-1}(1 - 2a_{n-1}ax_{n-1}) + a_{n-1}\left(\alpha_{n-1}\frac{\epsilon_{n-1}^{+} - \epsilon_{n-1}^{-}}{2c_{n-1}} + (1-\alpha_{n-1})\delta_{n-1}\right).
\end{aligned}
$$

The recursions can be rewritten in the general form

$$x_n = x_{n-1}B_{n-1} + C_{n-1},$$

where $B_{n-1} = 1 - 2a_{n-1}a$ and C_{n-1} depends on the algorithm. Then

$$
\begin{aligned}
x_n &= x_{n-1}B_{n-1} + C_{n-1} \\
&= x_{n-2}B_{n-2}B_{n-1} + C_{n-2}B_{n-1} + C_{n-1} \\
&= x_{n-3}B_{n-3}B_{n-2}B_{n-1} + C_{n-3}B_{n-2}B_{n-1} + C_{n-2}B_{n-1} + C_{n-1} \\
&= x_0\Pi_{i=}^{n-1}B_i + \sum_{j=0}^{n-2} C_j \Pi_{k=j+1}^{n-2} B_k + C_{n-1}
\end{aligned}
$$

Then the MSE of the nth estimate can be written as

$$
\begin{aligned}
MSE &= E[(x_n - x^*)^2] = E[x_n^2] \\
&= E[(x_0\Pi_{i=0}^{n-1}B_i + \sum_{j=0}^{n-2} C_j\Pi_{k=j+1}^{n-2}B_k + C_{n-1})^2] \\
&= x_0^2\Pi_{i=0}^{n-1}B_i^2 + E[(\sum_{j=0}^{n-2} C_j\Pi_{k=j+1}^{n-2}B_k)^2] + E[C_{n-1}^2] \\
&\qquad + 2x_0\Pi_{i=0}^{n-1}B_iE[\sum_{j=0}^{n-2} C_j\Pi_{k=j+1}^{n-2}B_k] + 2x_0\Pi_{i=0}^{n-1}B_iE[C_{n-1}] \\
&\qquad + 2E[C_{n-1}\sum_{j=0}^{n-2} C_j\Pi_{k=j+1}^{n-2}B_k] \\
&= x_0^2\Pi_{i=0}^{n-1}B_i^2 + \sum_{j=0}^{n-2} E[C_j^2]\Pi_{k=j+1}^{n-2}B_k^2 + E[C_{n-1}^2],
\end{aligned}
$$

where

$$E[C_k^2] = \begin{cases} a_k^2\sigma_g^2 & \text{for } RM; \\ \dfrac{\sigma_f^2 a_k^2}{2c_k^2} & \text{for } KW; \\ \dfrac{a_k^2}{2}\left(\dfrac{\alpha_k^2\sigma_f^2}{c_k^2} + (1-\alpha_k)^2\sigma_g^2\right) & \text{for } STAR, \end{cases}$$

which concludes the proof. □

Proposition 3.2.6. *If $\alpha_k^2\sigma_f^2 < \sigma_g^2 c_k^2$ for all k, then $MSE_{STAR} < MSE_{RM}$. If $2\sigma_g^2 c_k^2 < \sigma_f^2$ for all k, then $MSE_{RM} < MSE_{KW}$. If $2\sigma_g^2 c_k^2 < \sigma_f^2$ and $\alpha_k^2\sigma_f^2 < \sigma_g^2 c_k^2$ for all k, then $MSE_{STAR} < MSE_{KW}$.*

Proof. From Proposition 3.2.5,

$$\begin{aligned} MSE_{STAR} < MSE_{RM} &\iff \frac{a_k^2}{2}\left(\frac{\alpha_k^2\sigma_f^2}{c_k^2} + (1-\alpha_k)^2\sigma_g^2\right) < \sigma_g^2 a_k^2 \\ &\iff \frac{\alpha_k^2\sigma_f^2}{c_k^2} + (1-\alpha_k)^2\sigma_g^2 < 2\sigma_g^2 \\ &\iff \alpha_k^2\sigma_f^2 + c_k^2(1-\alpha_k)^2\sigma_g^2 < 2\sigma_g^2 c_k^2 \\ &\iff \alpha_k^2\sigma_f^2 + (1-\alpha_k)^2\sigma_g^2 c_k^2 < 2\sigma_g^2 c_k^2 \\ &\iff \alpha_k^2\sigma_f^2 < (2-(1-\alpha_k)^2)\sigma_g^2 c_k^2 \\ &\iff \alpha_k^2\sigma_f^2 < \sigma_g^2 c_k^2, \\ MSE_{RM} < MSE_{KW} &\iff \sigma_g^2 a_k^2 < \frac{\sigma_f^2 a_k^2}{2c_k^2} \\ &\iff 2\sigma_g^2 c_k^2 < \sigma_f^2, \end{aligned}$$

for all k. If $\alpha_k^2\sigma_f^2 < \sigma_g^2 c_k^2$ and $2\sigma_g^2 c_k^2 < \sigma_f^2$, then $MSE_{STAR} < MSE_{KW}$.□

The exact MSE of the estimators resulting from each of the algorithms has three components. The first term is identical for all three algorithms and is the only component that contains the initial value, so it does not affect our comparison of

the MSE across algorithms for the unconstrained, quadratic case. From Proposition 3.2.6, if certain conditions are satisfied, the MSE resulting from STAR-SA for all steepness levels is less than the MSE resulting from RM and KW for the same number of iterations

3.2.3 Numerical Experiments

We conduct two sets of numerical experiments to compare the performance of the STAR-SA algorithm against the classical RM and KW methods under various combinations of noise levels σ_f and σ_g [11]. We implement all three algorithms and generate the finite-time MSE of the estimate x_N for quadratic functions of the form $f(x) = -ax^2$, $a > 0$, and $\Theta = [-50, 50]$. The gain sequence is chosen as $a_n = \theta_a(n+1)^{-1}$ and the finite difference perturbation sequence is $c_n = \theta_c(n+1)^{-1/4}$. To illustrate the performance of the proposed algorithm, we choose parameter values $\theta_a \in \{1, 10, 100\}$, $\theta_c \in \{0.1, 1.0\}$, and $N \in \{100, 1000, 10000\}$ as well as steepness level $a \in \{10^k | k = -3, -2.5, \ldots, 1.5, 2\}$ and initial value $x_0 \in \{-50 + 5k | k = 1, \ldots, 10\}$. In a practical setting, the noise level of the function and its gradient could differ, so we consider a variety of noise level combinations, $\sigma_f \in \{10^k | k = -3, \ldots, 1\}$ and $\sigma_g \in \{10^k | k = -3, \ldots, 1\}$.

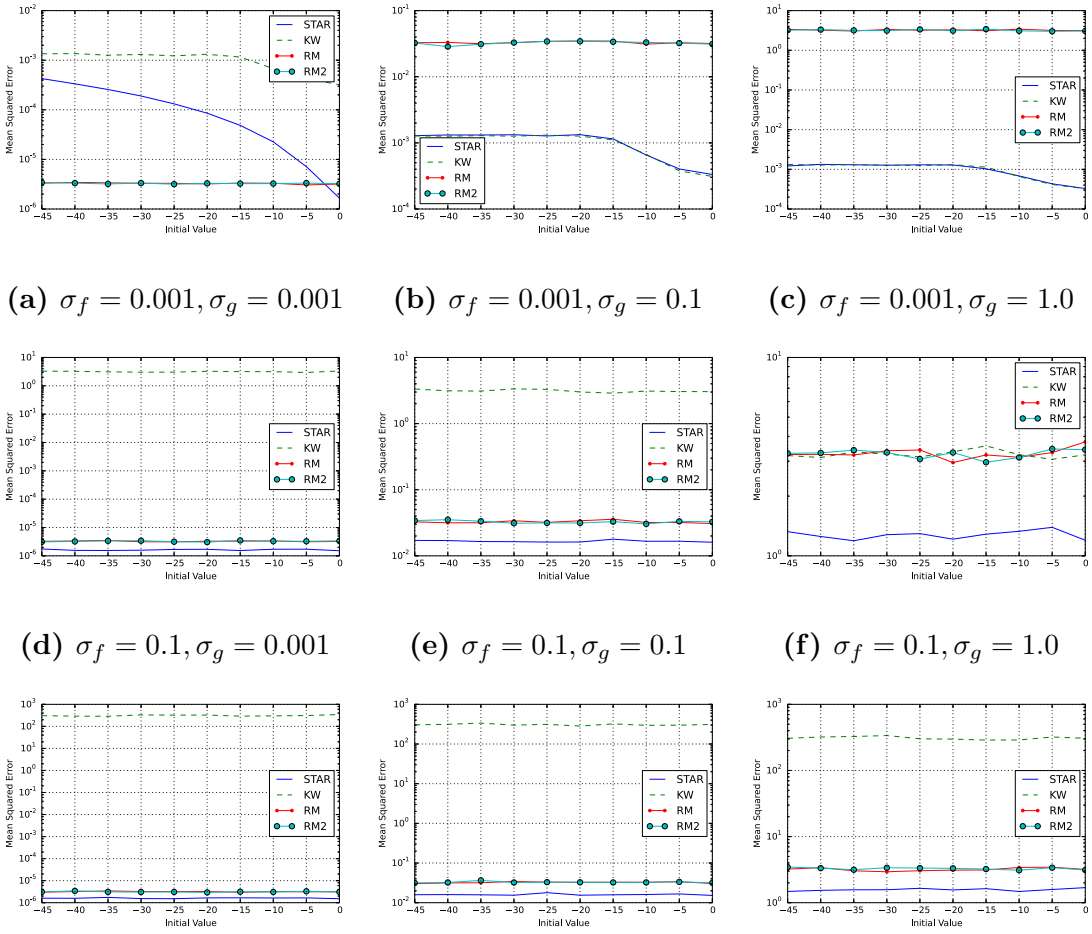
Although we implemented RM, KW, and STAR-SA for all of the parameter settings mentioned above, we will only discuss two representative subsets of our results. For both sets of numerical experiments, we choose $a_n = 10(n+1)^{-1}$ and $c_n = 0.1(n+1)^{-1/4}$, and $N = 1000$. The MSE of the 1000th iterate is computed for

the algorithms based on 2000 sample paths. In this section, MSE refers to the MSE of the 1000th iterate for STAR-SA and KW, and 2000th iteration for RM.

3.2.3.1 Experiment 1: vary initial value

For the first set of numerical experiments, we fix the steepness level and vary the initial value x_0 for different combinations of noise levels, σ_f and σ_g. We only present the results for $f(x) = -0.1x^2$ for $\sigma_f \in \{0.001, 0.1, 1.0\}$ and $\sigma_g \in \{0.001, 0.1, 1.0\}$, to illustrate the potential gains of STAR-SA. The STAR-SA algorithm outperforms KW and RM in six out of the nine combinations for all initial values, which can be seen in the last two rows of Figure 3.2 (i.e., 3.2d, 3.2e, 3.2f, 3.2g, 3.2h, and 3.2i). The commonality among these six cases is the higher function noise, i.e., $\sigma_f = 0.1, 1.0$. With the exception of Figure 3.2f, RM performs significantly better than KW, but STAR-SA has the lowest MSE among the three SA algorithms. Figure 3.2f depicts a case where RM and KW both perform similarly with intertwining lines, but combining secant and tangents averaged gradients in STAR-SA results in a much lower MSE.

In two of the remaining three cases, STAR-SA does not necessarily outperform RM nor KW, but it performs as well as the better of the two algorithms, as seen in Figures 3.2b and 3.2c. The function noise level is very low and the gradient noise is higher, i.e., $\sigma_f = 0.001$ and $\sigma_g > \sigma_f$, which increases the accuracy of the indirect gradient and has the opposite effect on the direct gradient. Not surprisingly in the two cases, KW outperforms RM, but the performance of STAR-SA is on par with



(a) $\sigma_f = 0.001, \sigma_g = 0.001$ **(b)** $\sigma_f = 0.001, \sigma_g = 0.1$ **(c)** $\sigma_f = 0.001, \sigma_g = 1.0$

(d) $\sigma_f = 0.1, \sigma_g = 0.001$ **(e)** $\sigma_f = 0.1, \sigma_g = 0.1$ **(f)** $\sigma_f = 0.1, \sigma_g = 1.0$

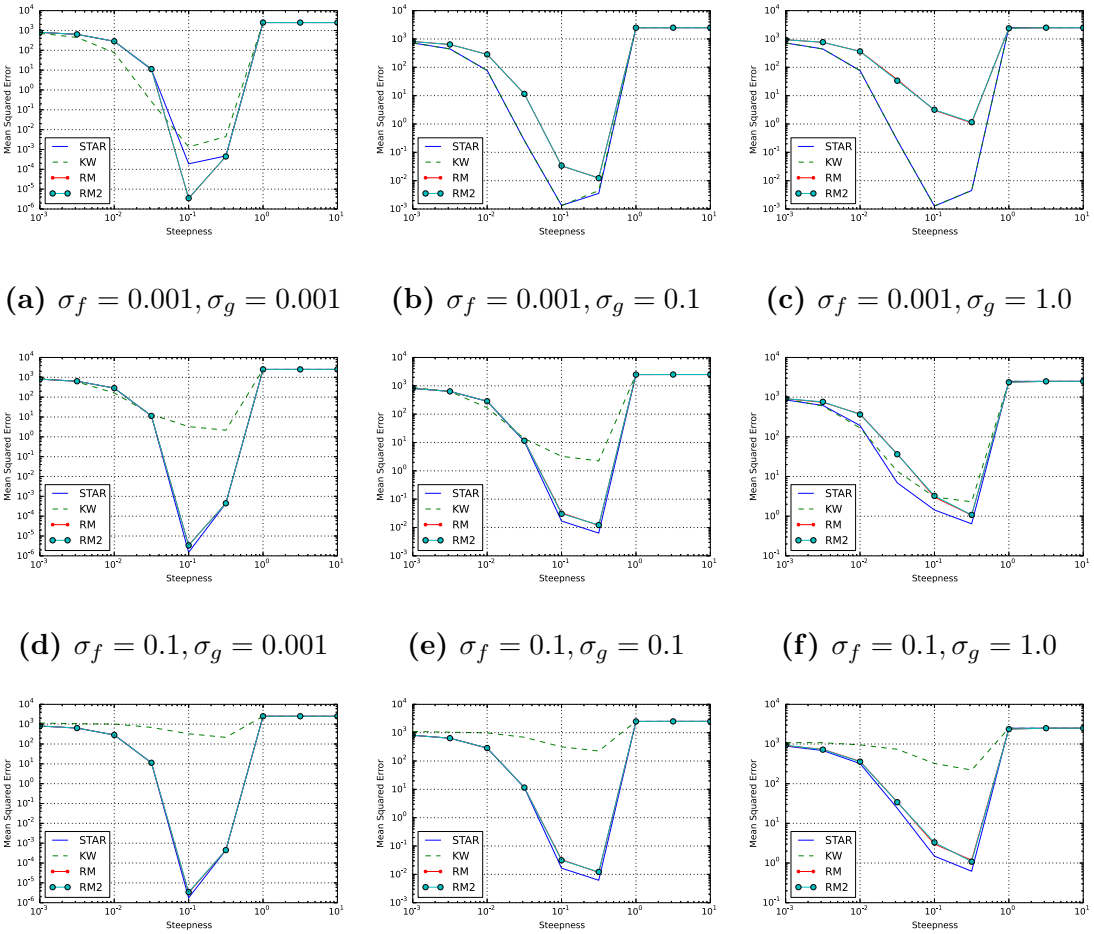
(g) $\sigma_f = 1.0, \sigma_g = 0.001$ **(h)** $\sigma_f = 1.0, \sigma_g = 0.1$ **(i)** $\sigma_f = 1.0, \sigma_g = 1.0$

Figure 3.2: $f(x) = -0.1x^2, \Theta = [-50, 50], a_n = 10(n+1)^{-1}, c_n = 0.1(n+1)^{-1/4}, N = 1000.$

KW. From the eight figures discussed, it appears that integrating both indirect and direct gradients together will either improve the SA performance, which can be significant, or in the worst case, it will not hurt the performance. 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 in the case when both noise levels are very low, i.e., $\sigma_f = \sigma_g = 0.001$, which is shown in Figure 3.2a. In this instance, RM performs better than STAR-SA with the exception of when the initial value is close to the optimum $x^* = 0$. In fact, the MSE of STAR-SA decreases as initial value x_0 approaches x^*.

3.2.3.2 Experiment 2: vary steepness level

In practice, the geometry of the function is unknown, so our second set of numerical experiments investigates the performance of the STAR-SA algorithm compared to RM and KW as a function of the steepness level a for different combinations of noise. We set the initial value to be far from the optimum and consider identical parameters as in the first experiment, i.e., $x_0 = -30, a_n = 10(n+1)^{-1}, c_n = 0.1(n+1)^{-1/4}$, and vary the steepness level a. We implement RM, KW, and STAR-SA and compute the MSE for $N = 1000$ and all combinations of $\sigma_f \in \{0.001, 0.1, 1.0\}$ and $\sigma_g \in \{0.001, 0.1, 1.0\}$, illustrated in Figure 3.3. For a fixed step size $\{a_n\}$, the MSE is much higher for both flat and steep quadratic functions across algorithms, which is most likely because the selected step size $\{a_n\}$ is too small for extremely flat and too large for steep functions, respectively. If the step size is not chosen



(a) $\sigma_f = 0.001, \sigma_g = 0.001$ **(b)** $\sigma_f = 0.001, \sigma_g = 0.1$ **(c)** $\sigma_f = 0.001, \sigma_g = 1.0$

(d) $\sigma_f = 0.1, \sigma_g = 0.001$ **(e)** $\sigma_f = 0.1, \sigma_g = 0.1$ **(f)** $\sigma_f = 0.1, \sigma_g = 1.0$

(g) $\sigma_f = 1.0, \sigma_g = 0.001$ **(h)** $\sigma_f = 1.0, \sigma_g = 0.1$ **(i)** $\sigma_f = 1.0, \sigma_g = 1.0$

Figure 3.3: $f(x) = -ax^2, \Theta = [-50, 50], a_n = 10(n+1)^{-1}, c_n = 0.1(n+1)^{-1/4}, N = 1000.$

appropriately, all three algorithms perform equally poorly, with the exception of the case when both noise levels are low, i.e., $\sigma_f = \sigma_g = 0.001$, where KW results in a lower MSE. The fixed step size and stopping time only appear to be appropriate for $a = 10^{-1}, 10^{-1/2}$. For all noise levels under the these two steepness levels, the MSE of STAR-SA is either less than or equal to RM and KW. The only instance when combining direct and indirect gradients is inferior is when both noise levels are low, i.e., $\sigma_f = \sigma_g = 0.001$ for $a = 10^{-1}$. Otherwise, STAR-SA shows promise since it either outperforms or at least it never performs worse than either RM or KW. When the variance of the function is high, i.e., $\sigma_f \in \{0.1, 1.0\}$, KW performs poorly across steepness levels.

3.2.3.3 Results Summary

The following conclusions are specifically for the fixed parameter setting $f(x) = -ax^2, a_n = 10n^{-1}, c_n = 0.1n^{-1/4}$.

- If the function is neither too steep nor too flat:
  - All algorithms are insensitive to the initial start value.
  - If σ_f is large, STAR-SA has a significantly lower MSE than that of KW.
  - If σ_f and σ_g are high, the MSE of STAR-SA is lower than that of RM, but the difference is not as prominent as the difference compared to that of KW.
- For very flat functions:
  - If $\sigma_f < \sigma_g$, then $MSE_{STAR} \approx MSE_{KW}$ and $MSE_{KW} < MSE_{RM}$, and this gap decreases as the function flattens.

- If $\sigma_g < \sigma_f$ or if σ_f and σ_g are both high, then $MSE_{STAR} \approx MSE_{RM}$ and $MSE_{RM} < MSE_{KW}$, and this difference shrinks as the steepness level decreases.

- For very steep functions, the performances of the algorithms are equally poor.

Overall, as a result from both sets of numerical tests, the STAR-SA algorithm either performs significantly better than both KW and RM or the MSE is approximately equal to that of the algorithm with the lower MSE. Therefore, the new STAR-SA algorithm is competitive against the classic KW and RM methods.

3.3 STAR-SPSA

Now we extend STAR-SA to the multidimensional case. The direct extension using finite differences is computationally inefficient for high-dimensional problems, so instead, we consider the SP gradient, which only requires the two performance estimates $\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n)$ and $\tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)$ for each indirect gradient (**S**ecant). The available direct gradient estimates $\nabla\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n)$ and $\nabla\tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)$ can be averaged to provide another gradient estimate (**T**angents **A**ve**R**aged).

For multivariate stochastic optimization problems, we present the **S**ecant-**T**angents **A**ve**R**aged simultaneous perturbation stochastic approximation (STAR-SPSA) gradient estimate, which is a convex combination of an SP gradient estimate and an average of two associated direct gradients. For each dimension $i \in \{1, \ldots, d\}$,

the ith component of the STAR-SPSA gradient can be written as

$$g_{STAR,i}(\mathbf{x}_n) = \alpha_{n,i}\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}} + (1-\alpha_{n,i})\frac{\nabla\tilde{f}_i(\mathbf{x}_n + c_n\mathbf{\Delta}_n) + \nabla\tilde{f}_i(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2}, \quad (3.17)$$

where $\mathbf{\Delta}_n = (\Delta_{n,1}, \ldots, \Delta_{n,d})$, $\alpha_{n,i} \in [0,1]$, and $c_n \in \mathbb{R}^+$ for $n \in \mathbb{N}$, and $c_n \to 0$ as $n \to \infty$.

Let $\tilde{f}(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n) = f(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n) + \epsilon_n^{\pm}$, $\nabla\tilde{f}_i(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n) = \nabla f_i(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n) + \delta_{n,i}^{\pm}$, where $\epsilon_n^{\pm}$ and $\delta_{n,i}^{\pm}$ are Gaussian noise with zero mean for $i = 1, \ldots, d$. For notational simplicity, let $Var(\tilde{f}(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n)) = \sigma_f^2$, $Var(\nabla\tilde{f}_i(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n)) = \sigma_{g,i}^2$, $Corr(\tilde{f}(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n), \nabla\tilde{f}_i(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n)) = \rho_i$, $Var(g_{SPSA,i}(\mathbf{x}_n)) = Var\left(\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}}\right) = \frac{\sigma_f^2}{2c_n^2\Delta_{n,i}^2}$, $Var(g_{TAR,i}(\mathbf{x}_n)) = Var\left(\frac{\nabla\tilde{f}_i(\mathbf{x}_n + c_n\mathbf{\Delta}_n) + \nabla\tilde{f}_i(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2}\right) = \frac{\sigma_{g,i}^2}{2}$, and $Var(g_{STAR,i}(\mathbf{x}_n)) = \sigma_{n,i}^2$ for $i = 1, \ldots, d$.

Analogous to Lemma 3.2.1 and Proposition 3.2.2, the next two results provide a STAR-SPSA gradient variance minimizing deterministic weight sequence that guarantees the variance of the STAR-SPSA gradient is strictly less than that of RM and SPSA.

3.3.1 Optimal Deterministic Weights

Lemma 3.3.1. *For g_{STAR} defined in (3.17), assume $\epsilon_n^+ \perp \epsilon_n^-$ and $\delta_{n,i}^+ \perp \delta_{n,i}^-$ for all n and $i = 1, \ldots, d$. Then $Var(g_{STAR,i}(\mathbf{x}_n))$ is minimized when*

$$\alpha_{n,i}^* = \frac{\sigma_{g,i}^2 c_n^2 \Delta_{n,i}^2}{\sigma_f^2 + \sigma_{g,i}^2 c_n^2 \Delta_{n,i}^2}. \quad (3.18)$$

Proof. For notational simplicity, let $\tilde{f}_n^{\pm} = \tilde{f}(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n)$, $\tilde{\mathbf{g}}_n^{\pm} = \nabla\tilde{f}(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n)$, and $\tilde{\mathbf{g}}_n^{\pm} = (\tilde{g}_{n,1}^{\pm}), \ldots, \tilde{g}_{n,d}^{\pm})$.

By the orthogonality assumption, the ith component of the variance can be written as

$$\begin{aligned}
\sigma_{n,i}^2 &= Var\left(\alpha_{n,i}\frac{\tilde{f}_n^+ - \tilde{f}_n^-}{2c_n\Delta_{n,i}} + (1-\alpha_{n,i})\frac{\tilde{g}_{n,i}^+ + \tilde{g}_{n,i}^-}{2}\right) \\
&= Var\left(\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\tilde{f}_n^+ + \frac{1-\alpha_{n,i}}{2}\tilde{g}_{n,i}^+\right) + Var\left(-\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\tilde{f}_n^- + \frac{1-\alpha_{n,i}}{2}\tilde{g}_{n,i}^-\right) \\
&= \left(\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\right)^2 Var(\tilde{f}_n^+) + \left(\frac{1-\alpha_{n,i}}{2}\right)^2 Var(\tilde{g}_{n,i}^+) + 2\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\frac{1-\alpha_{n,i}}{2}Cov(\tilde{f}_n^+, \tilde{g}_{n,i}^+) \\
&\quad + \left(\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\right)^2 Var(\tilde{f}_n^-) + \left(\frac{1-\alpha_{n,i}}{2}\right)^2 Var(\tilde{g}_{n,i}^-) - 2\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\frac{1-\alpha_{n,i}}{2}Cov(\tilde{f}_n^-, \tilde{g}_{n,i}^-) \\
&= \left(\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\right)^2 \sigma_f^2 + \left(\frac{1-\alpha_{n,i}}{2}\right)^2 \sigma_{g,i}^2 + 2\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\frac{1-\alpha_{n,i}}{2}\rho_i\sigma_f\sigma_{g,i} \\
&\quad + \left(\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\right)^2 \sigma_f^2 + \left(\frac{1-\alpha_{n,i}}{2}\right)^2 \sigma_{g,i}^2 - 2\frac{\alpha_{n,i}}{2c_n\Delta_{n,i}}\frac{1-\alpha_{n,i}}{2}\rho_i\sigma_f\sigma_{g,i} \\
&= \frac{\alpha_{n,i}^2}{2c_n^2\Delta_{n,i}^2}\sigma_f^2 + \frac{(1-\alpha_{n,i})^2}{2}\sigma_{g,i}^2, \qquad (3.19)
\end{aligned}$$

where the second equality follows from the independence of the function/gradient at different points. Differentiate the variance (3.19) with respect to the weight $\alpha_{n,i}$ to obtain

$$\frac{\partial\sigma_{n,i}^2}{\partial\alpha_{n,i}} = \frac{\alpha_{n,i}}{c_n^2\Delta_{n,i}^2}\sigma_f^2 - (1-\alpha_{n,i})\sigma_{g,i}^2. \qquad (3.20)$$

Solve for the zero of (3.20) to obtain (3.18), which is a global minimum since (3.20) is monotonically increasing in $\alpha_{n,i}$. □

Proposition 3.3.2. *Let $\widehat{\nabla} f(\mathbf{x}_n)$ be defined as in (3.17) and the convex weight be defined in (3.18). If the conditions of Lemma 3.3.1 hold, then $Var(g_{STAR,i}(\mathbf{x}_n)) \leq \min\{Var(g_{SPSA,i}(\mathbf{x}_n)), Var(g_{TAR,i}(\mathbf{x}_n))\}$ for $i = 1, \ldots, d$ and all n, where the inequality is strict if $0 < \sigma_f^2$ and $0 < \sigma_{g,i}^2 c_n^2 \Delta_{n,i}^2$ hold.*

Proof. By the conditions of Lemma 3.3.1, the ith component of the variance

can be written as

$$\sigma_{n,i}^2 = \frac{\alpha_{n,i}^2}{2c_n^2\Delta_{n,i}^2}\sigma_f^2 + \frac{(1-\alpha_{n,i})^2}{2}\sigma_{g,i}^2. \tag{3.21}$$

Substitute $\alpha_{n,i}^*$ from (3.18) into the variance (3.21) to obtain

$$\begin{aligned}\sigma_{n,i}^2 &= \frac{1}{2c_n^2\Delta_{n,i}^2}\left(\frac{\sigma_{g,i}^2c_n^2\Delta_{n,i}^2}{\sigma_f^2+\sigma_{g,i}^2c_n^2\Delta_{n,i}^2}\right)^2\sigma_f^2 + \frac{1}{2}\left(1-\frac{\sigma_{g,i}^2c_n^2\Delta_{n,i}^2}{\sigma_f^2+\sigma_{g,i}^2c_n^2\Delta_{n,i}^2}\right)^2\sigma_{g,i}^2 \\ &= \frac{\sigma_f^2\sigma_{g,i}^2}{2\left(\sigma_f^2+\sigma_{g,i}^2c_n^2\Delta_{n,i}^2\right)}.\end{aligned}$$

To show $\sigma_{n,i}^2$ is strictly less than $\min\{Var(g_{SPSA,i}(\mathbf{x}_n)), Var(g_{TAR,i}(\mathbf{x}_n))\}$, we consider two cases: 1) $\sigma_{n,i}^2 < \frac{\sigma_f^2}{2c_n^2\Delta_{n,i}^2}$ and 2) $\sigma_{n,i}^2 < \frac{\sigma_{g,i}^2}{2}$. However, since $c_n \to 0$ as $n \to \infty$, then $\frac{\sigma_f^2}{2c_n^2\Delta_{n,i}^2} \to \infty$, so eventually, we will be in the latter case.

Case 1: If $\sigma_{n,i}^2 < \frac{\sigma_f^2}{2c_n^2\Delta_{n,i}^2}$, then $\frac{\sigma_f^2\sigma_{g,i}^2}{2\left(\sigma_f^2+\sigma_{g,i}^2c_n^2\Delta_{n,i}^2\right)} < \frac{\sigma_f^2}{2c_n^2\Delta_{n,i}^2}$, which holds when $0 < \sigma_f^2$.

Case 2: If $\sigma_{n,i}^2 < \frac{\sigma_{g,i}^2}{2}$, then $\frac{\sigma_f^2\sigma_{g,i}^2}{2\left(\sigma_f^2+\sigma_{g,i}^2c_n^2\Delta_{n,i}^2\right)} < \frac{\sigma_{g,i}^2}{2}$, which holds when $0 < \sigma_{g,i}^2c_n^2\Delta_{n,i}^2$.

If $0 < \sigma_f^2$ and $0 < \sigma_{g,i}^2c_n^2\Delta_{n,i}^2$, then the strict inequality holds; however if either $\sigma_f = 0$ or $\sigma_{g,i}^2c_n^2\Delta_{n,i}^2 = 0$, then the same argument holds for the two cases with " $<$ " replaced with " $\leq$," resulting in a non-strict inequality. Therefore, $\sigma_{n,i}^2 \leq \min\left\{\frac{\sigma_f^2}{2c_n^2\Delta_{n,i}^2}, \frac{\sigma_{g,i}^2}{2}\right\} = \min\{Var(g_{SPSA,i}(\mathbf{x}_n)), Var(g_{TAR,i}(\mathbf{x}_n))\}$, where the inequality is strict for $0 < \sigma_f^2$ and $0 < \sigma_{g,i}^2c_n^2\Delta_{n,i}^2$. □

As in the STAR-SA result, the weight $\alpha_{n,i}^*$ for STAR-SPSA is also independent of the correlation coefficient because of the homogeneous noise

3.3.2 Convergence

We define the error and bias of the STAR-SPSA gradient estimate respectively as

$$\begin{aligned}\mathbf{b}_n(\mathbf{x}_n) &= E[g_{STAR}(\mathbf{x}_n) - \nabla f(\mathbf{x}_n)|\mathbf{x}_n], \\ \mathbf{e}_n(\mathbf{x}_n) &= g_{STAR}(\mathbf{x}_n) - E[g_{STAR}(\mathbf{x}_n)|\mathbf{x}_n],\end{aligned}$$

where $\mathbf{b}_n(\mathbf{x}_n) = (b_{n,1}(\mathbf{x}_n), \ldots, b_{n,d}(\mathbf{x}_n))$ and $\mathbf{e}_n(\mathbf{x}_n) = (e_{n,1}(\mathbf{x}_n), \ldots, e_{n,d}(\mathbf{x}_n))$.

The following lemma is used in the convergence proof of Theorem 3.3.4.

Lemma 3.3.3. *Suppose $\{\mathbf{x}_n\}$ is generated under recursion (1.2) with $\Theta = \mathbb{R}^d$ using the STAR-SPSA gradient estimate defined in (3.17). If $\{\Delta_{n,i}\}_{i=1}^d$ are i.i.d., symmetrically distributed with mean zero, uniformly bounded with finite inverse moments, and $f(\cdot)$ is three-times continuously differentiable and $f^{(3)}(\cdot)$ is uniformly bounded, then*

$$\mathbf{b}_n(\mathbf{x}_n) \to \mathbf{0} \text{ as } n \to \infty.$$

Proof. We use proof techniques similar to those in [55] by applying results from [39].

By assumption for each $n \in \mathbb{N}$, there exist positive constants K_1, K_2, and K_3 such that $|\Delta_{n,l}| \leq K_1$ a.s., $E[|\Delta_{n,l}^{-1}|] \leq K_2$ for $l = 1, \ldots, d$ and $|f_{i,j,k}^{(3)}(\cdot)| \leq K_3 \ \forall i, j, k$.

$$\begin{aligned}b_{n,i}(\mathbf{x}_n) &= E[\alpha_{n,i} \cdot g_{SPSA,i}(\mathbf{x}_n) + (1 - \alpha_{n,i}) \cdot g_{TAR,i}(\mathbf{x}_n) - \nabla f_i(\mathbf{x}_n)|\mathbf{x}_n] \\ &= \alpha_{n,i} E[g_{SPSA,i}(\mathbf{x}_n) - \nabla f_i(\mathbf{x}_n)|\mathbf{x}_n] + (1 - \alpha_{n,i}) E[g_{TAR,i}(\mathbf{x}_n) - \nabla f_i(\mathbf{x}_n)|\mathbf{x}_n].\end{aligned}$$

Since $E[\epsilon_n^+ - \epsilon_n^-|\mathbf{x}_n] = 0$ a.s. and $f^{(3)}(\cdot)$ exists and is continuous, using Taylor's

series expansions, we have

$$f(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n) = f(\mathbf{x}_n) \pm c_n\langle f'(\mathbf{x}_n), \mathbf{\Delta}_n\rangle + c_n^2\langle f''(\mathbf{x}_n)\mathbf{\Delta}_n, \mathbf{\Delta}_n\rangle \\ \pm c_n^3 f^{(3)}(\mathbf{t}_n^{\pm})[\mathbf{\Delta}_n \otimes \mathbf{\Delta}_n \otimes \mathbf{\Delta}_n], \quad (3.22)$$

where $\mathbf{t}_n^+$ and $\mathbf{t}_n^-$ are on the line segment between $\mathbf{x}_n$ and $\mathbf{x}_n \pm c_n\mathbf{\Delta}_n$, respectively, and $\otimes$ denotes the Kronecker product. Using (3.22), we have

$$\begin{aligned} & E[g_{SPSA,i}(\mathbf{x}_n) - \nabla f_i(\mathbf{x}_n)|\mathbf{x}_n] \\ &= E\left[\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}} - \Delta_{n,i}^{-1}\langle f'(\mathbf{x}_n), \mathbf{\Delta}_n\rangle \Big| \mathbf{x}_n\right] \\ &= E\left[\frac{f(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - f(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}} - \Delta_{n,i}^{-1}\langle f'(\mathbf{x}_n), \mathbf{\Delta}_n\rangle \Big| \mathbf{x}_n\right] \\ &= \frac{c_n^2}{12}E\left[\Delta_{n,i}^{-1}(f^{(3)}(\mathbf{t}_n^+) + f^{(3)}(\mathbf{t}_n^-))[\mathbf{\Delta}_n \otimes \mathbf{\Delta}_n \otimes \mathbf{\Delta}_n]|\mathbf{x}_n\right]. \quad (3.23) \end{aligned}$$

The magnitude of the right hand side of (3.23) is bounded by

$$\frac{c_n^2 K_3}{6}\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} E\Big|\frac{\Delta_{n,i}\Delta_{n,j}\Delta_{n,k}}{\Delta_{n,i}}\Big| \\ \leq \frac{c_n^2 K_3}{6}\cdot\{[d^3 - (d-1)^3]K_1^2 + (d-1)^3 K_2 K_1^3\}. \quad (3.24)$$

Similarly, since $E[\delta_{n,i}^+ + \delta_{n,i}^-|\mathbf{x}_n] = 0$ a.s. for $i = 1, \ldots, d$, by Taylor's series expansions,

$$\nabla f_i(\mathbf{x}_n \pm c_n\mathbf{\Delta}_n) = \nabla f_i(\mathbf{x}_n) \pm c_n\langle\nabla^2 f_i(\mathbf{x}_n), \mathbf{\Delta}_n\rangle + c_n^2\langle\nabla^3 f_i(\mathbf{s}_n^{\pm})\mathbf{\Delta}_n, \mathbf{\Delta}_n\rangle, \quad (3.25)$$

where $\mathbf{s}_n^+$ and $\mathbf{s}_n^-$ are on the line segment between $\mathbf{x}_n$ and $\mathbf{x}_n \pm c_n\mathbf{\Delta}_n$, respectively.

Using (3.25),

$$\begin{aligned} & E[g_{TAR,i}(\mathbf{x}_n) - \nabla f_i(\mathbf{x}_n)|\mathbf{x}_n] \\ & \quad = E\left[\frac{\nabla \tilde{f}_i(\mathbf{x}_n + c_n\boldsymbol{\Delta}_n) + \nabla \tilde{f}_i(\mathbf{x}_n - c_n\boldsymbol{\Delta}_n)}{2} - \nabla f_i(\mathbf{x}_n)\Big|\mathbf{x}_n\right] \\ & \quad = E\left[\frac{\nabla f_i(\mathbf{x}_n + c_n\boldsymbol{\Delta}_n) + \nabla f_i(\mathbf{x}_n - c_n\boldsymbol{\Delta}_n)}{2} - \nabla f_i(\mathbf{x}_n)\Big|\mathbf{x}_n\right] \\ & \quad = \frac{c_n^2}{2} E\left[\boldsymbol{\Delta}_n^T (f^{(3)}(\mathbf{s}_n^+) + f^{(3)}(\mathbf{s}_n^-))\boldsymbol{\Delta}_n|\mathbf{x}_n\right]. \end{aligned} \tag{3.26}$$

The magnitude of the right hand side of (3.26) is bounded by

$$c_n^2 K_3 \sum_{i=1}^{d} E|\Delta_{n,i}^2| \leq c_n^2 K_3 d K_1^2. \tag{3.27}$$

Apply bounds (3.24) and (3.27) to obtain

$$\begin{aligned} b_{n,i}(\mathbf{x}_n) \quad \leq \quad & (1-\alpha_{n,i}) c_n^2 K_3 d^2 K_1^2 \\ & + \frac{\alpha_{n,i} c_n^2 K_3}{6} \cdot \{[d^3 - (d-1)^3] K_1^2 + (d-1)^3 K_2 K_1^3\}. \end{aligned} \tag{3.28}$$

The RHS of (3.28) converges to zero since $c_n \to 0$ as $n \to \infty$, and this holds for all i, which concludes the proof. □

Theorem 3.3.4. *Let $\{\mathbf{x}_n\}$ be a sequence generated using recursion (1.2) with $\Theta = \mathbb{R}^d$ using the STAR-SPSA gradient estimate defined in (3.17). Assume the conditions of Lemma 3.3.3 in addition to the following hold:*

1. *There exist positive sequences $\{a_n\}$ and $\{c_n\}$ such that $c_n \to 0$ as $n \to \infty$, $\sum_{n=1}^{\infty} a_n = \infty$, $\sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty$.*

2. *There exist positive constants C_1, C_2, C_3 such that $E[\nabla \tilde{f}_i(\mathbf{x}_n \pm c_n\boldsymbol{\Delta}_n)^2] \leq C_1$, $E[\tilde{f}(\mathbf{x}_n \pm c_n\boldsymbol{\Delta}_n)^4] \leq C_2^2$, and $E[\Delta_{n,i}^{-4}] \leq C_3^2$ for $i = 1, \ldots, d$.*

3. *For all* $n, ||\mathbf{x}_n|| < \infty$ *a.s.*

4. *Let* $\mathbf{x}^*$ *be an asymptotically stable solution to the ODE* $\partial \mathbf{x}(t)/\partial t = -\nabla f(\mathbf{x})$.

5. *There exists a compact set* $D \subseteq D(\mathbf{x}^*)$ *such that* $\mathbf{x}_n \in D$ *infinitely often, where* $D(\mathbf{x}^*)$ *is the domain of attraction (i.e.,* $D(\mathbf{x}^*) = \{\mathbf{x}_0 | \lim_{t\to\infty} \mathbf{x}(t|\mathbf{x}_0) = \mathbf{x}^*\}$*).*

Then $\mathbf{x}_n \to \mathbf{x}^*$ *a.s. as* $n \to \infty$.

Proof. By Lemma 2.2.1 and Theorem 2.3.1 in [39], if conditions 1 - 5 are satisfied, then $\mathbf{x}_n \to \mathbf{x}^*$ a.s. as $n \to \infty$ if

(a) $||\mathbf{b}_n(\mathbf{x}_n)|| < \infty$ for all n and $\mathbf{b}_n(\mathbf{x}_n) \to 0$ as $n \to \infty$ a.s.,

(b) $\lim_{n\to\infty} P\left(\sup_{m\geq n} || \sum_{k=n}^{m} a_k \mathbf{e}_k(\mathbf{x}_k)|| \geq \eta\right) = 0$ for all $\eta > 0$.

Since (a) follows directly from Lemma 3.3.3, it remains to establish (b). Apply Doob's inequality to the martingale sequence $\{\sum_{i=k}^{m} a_i \mathbf{e}_i(\mathbf{x}_i)\}_{m\geq k}$ to obtain

$$P\left(\sup_{m\geq n} || \sum_{k=n}^{m} a_k \mathbf{e}_k|| \geq \eta\right) \leq \eta^{-2} E\left[|| \sum_{k=n}^{m} a_k \mathbf{e}_k||^2\right] = \eta^{-2} \sum_{k=n}^{m} a_k^2 E||\mathbf{e}_k||^2. \quad (3.29)$$

Consider $i \in \{1, \ldots, d\}$. Then the variance of the ith gradient component can be expressed as

$$\begin{aligned} E[g_{STAR,i}^2(\mathbf{x}_n)] &= \alpha_{n,i}^2 E[g_{SPSA,i}^2(\mathbf{x}_n)] + 2\alpha_{n,i}(1-\alpha_{n,i}) E[g_{TAR,i}(\mathbf{x}_n) \cdot g_{SPSA,i}(\mathbf{x}_n)] \\ &\quad + (1-\alpha_{n,i})^2 E[g_{TAR,i}^2(\mathbf{x}_n)]. \quad (3.30) \end{aligned}$$

It remains to show that each of the three terms on the right hand side of (3.30) are bounded. We apply $|a+b|^r \leq 2^{r-1}(|a|^r + |b|^r)$ and Holder's inequality to obtain

$$\begin{aligned} E[g_{TAR,i}^2(\mathbf{x}_n)] &= E\left[\left(\frac{\nabla \tilde{f}_i(\mathbf{x}_n + c_n \boldsymbol{\Delta}_n) + \nabla \tilde{f}_i(\mathbf{x}_n - c_n \boldsymbol{\Delta}_n)}{2}\right)^2\right] \\ &\leq \frac{1}{2} E\left[(\nabla \tilde{f}_i(\mathbf{x}_n + c_n \boldsymbol{\Delta}_n)^2 + (\nabla \tilde{f}_i(\mathbf{x}_n - c_n \boldsymbol{\Delta}_n)^2\right] \leq C_1 \quad (3.31) \end{aligned}$$

and

$$\begin{aligned}
E[g^2_{SPSA,i}(\mathbf{x}_n)] &= E\left[\left(\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}}\right)^2\right] \\
&\leq \frac{1}{2c_n^2}E\left[\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n)^2 + \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)^2}{\Delta^2_{n,i}}\right] \\
&\leq \frac{1}{2c_n^2}\left[E\left[|(\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n)|^4\right]^{1/2} + E\left[|(\tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)|^4\right]^{1/2}\right] \cdot E\left[\left|\frac{1}{\Delta_{n,i}}\right|^4\right]^{1/2} \\
&\leq \frac{C_2C_3}{c_n^2}. \qquad (3.32)
\end{aligned}$$

Then we apply Jensen's inequality, (3.31), and (3.32) to obtain

$$\begin{aligned}
&E[g_{TAR,i}(\mathbf{x}_n) \cdot g_{SPSA,i}(\mathbf{x}_n)] \\
&= E\left[\left(\frac{\nabla\tilde{f}_i(\mathbf{x}_n + c_n\mathbf{\Delta}_n) + \nabla\tilde{f}_i(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2}\right) \cdot \left(\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}}\right)\right] \\
&\leq E\left[\left|\frac{\nabla\tilde{f}_i(\mathbf{x}_n + c_n\mathbf{\Delta}_n) + \nabla\tilde{f}_i(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2}\right|\right] \cdot E\left[\left|\frac{\tilde{f}(\mathbf{x}_n + c_n\mathbf{\Delta}_n) - \tilde{f}(\mathbf{x}_n - c_n\mathbf{\Delta}_n)}{2c_n\Delta_{n,i}}\right|\right] \\
&\leq \frac{\sqrt{C_1C_2C_3}}{c_n}. \qquad (3.33)
\end{aligned}$$

Applying (3.31), (3.32), and (3.33) yields

$$E[||e_n||^2] \leq d\left(\frac{C_2C_3}{4c_n^2} + \frac{\sqrt{C_1C_2C_3}}{2c_n} + \frac{C_1}{4}\right). \qquad (3.34)$$

By condition 1, (3.34), and (3.29), (b) holds. □

3.3.3 Numerical Experiments

3.3.3.1 9-station Closed Jackson Network

For the multidimensional case, we implement RM, SPSA, and STAR-SPSA on a 9-station closed Jackson queueing network problem from [35], depicted in Figure 3.4 with associated transition probabilities. In a closed Jackson queueing network,

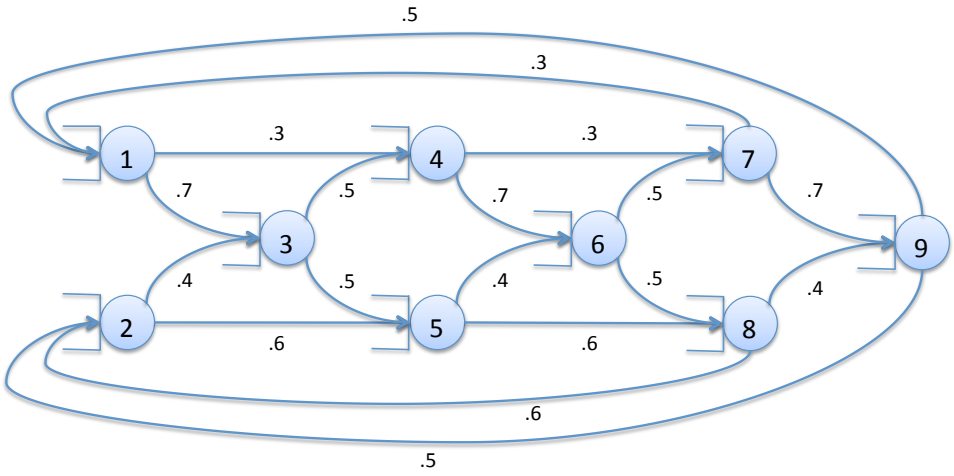


Figure 3.4: 9-station closed Jackson queueing network.

the number of customers in the system remains fixed, hence the absence of arrows from outside.

The objective is to maximize the total throughput of the system $TP(\mathbf{x})$ (total # of customers served/total time), given restrictions on the mean service time $x^{(i)}, i = 1, \ldots, d$, which can be summarized as

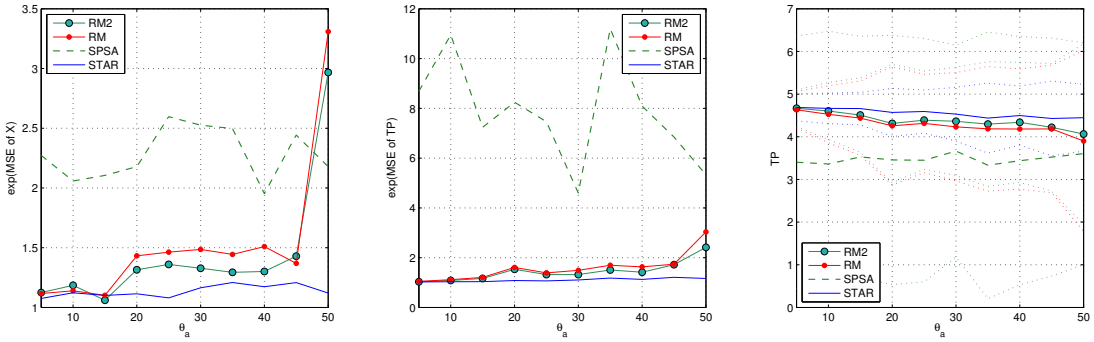
$$\begin{aligned} \max_{\mathbf{x}\in\Theta} \quad & TP(\mathbf{x}) \\ \text{s.t.} \quad & \sum_{i=1}^{d} x^{(i)} = M, \\ & x^{(i)} > 0 \text{ for } i = 1, \ldots, d, \end{aligned}$$

where $M = 10$, $d = 9$ (number of stations). The network consists of 9 first-come, first-served stations, where the service time follows an exponential distribution with mean service time $x^{(i)}$ for station i and with 10 customers in the system.

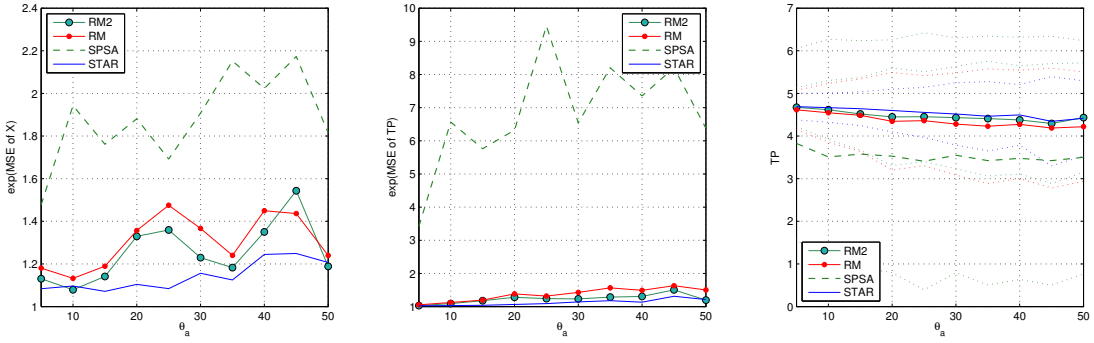
The (equality and positivity) constraints prevent straightforward implemen-

tations of the gradients and projection operator. For the direct gradient, we use the IPA gradient estimate designed specifically for the total throughput of a closed Jackson queueing network proposed in [34], with a slight modification by rewriting one of the estimates as a function of the other components and using the chain rule to take into account the equality constraint (i.e., $x^{(r)} = M - \sum_{i=1,i\neq r}^{d} x^{(i)}$, where r is an arbitrary integer between 1 and d, inclusive). For the SP gradient estimator, the components in the random vector $\boldsymbol{\Delta}_n$ must sum to zero to ensure the perturbed components satisfy the equality condition (i.e., $\Delta_{n,r} = 0$, $\Delta_{n,i} = \pm 1$ for $i \neq r$, where $\sum_{i=1,i\neq r}^{9} \Delta_{n,i} = 0$). In addition, each perturbed component must be positive, i.e., $x_n^{(i)} > c_n$ for all i. After each iteration, a projection must be applied to maintain feasibility; however, an orthogonal projection onto the hyperplane could violate the $x_n^{(i)} > c_n$ condition for some i, so we make a slight adjustment by projecting the current iterate back to the previous iterate if the condition is violated.

In the SA algorithms, we use the parameters $a_n = \theta_a(n+1)^{-1}$, $\alpha_{n,i} = c_n^2/(1+c_n^2)$, uniform start values $x_0^{(i)} = 10/9$ for $i = 1, \ldots, 9$, $c_n = \theta_c(n+1)^{-1/4}$, 20 macroreplications, a stopping time of $N = 50$ for SPSA and STAR-SPSA, and $N = 100$ for RM. For simplicity, we drop the second subscript i from the convex weight $\alpha_{n,i}$, since they are identical for all i. The update occurs for only eight estimates since the rth component is a result of the binding constraint. However, in our preliminary experiments, the SA algorithms often performed poorly when r was fixed for each iteration, so instead, we let r be a uniform random integer between 1 and 9, inclusive. For each simulation run, we generate the throughput and gradient estimates using 300 customers, which is large enough for a valid non-zero direct



(a) $c_n = 0.1(n+1)^{-1/4}$



(b) $c_n = 0.3(n+1)^{-1/4}$

Figure 3.5: 9-station closed Jackson queueing network, $\sum_{i=1}^{9} x^{(i)} = 10$, $x^{(i)} > 0$, $a_n = \theta_a/(n+1)$, $\alpha_n = c_n^2/(1+c_n^2)$, $N = 50$, customers serviced $= 300$, macroreplications $= 20$.

gradient. If the direct IPA gradient is generated with an insufficient number of customers, the the gradient could be exactly zero or invalid since the denominator is zero.

We compute the total MSE of the estimates $x_N^{(i)}$ for $i = 1, \ldots, 9$ and MSE of the throughput $TP(\mathbf{x}_N)$ using $\mathbf{x}^* = (1.66, .68, .983, 1.66, .68, .983, 1.66, .68, .983)$ and $TP(\mathbf{x}^*) = 4.82$, as well as the mean throughput with a 95% confidence band using 20 macroreplications for a range of step sizes and two different perturbation sizes,

i.e., $\theta_a \in \{5k | k = 1, \ldots, 10\}$ and $\theta_c \in \{0.1, 0.3\}$. Figures 3.5a and 3.5b illustrate the performances as a function of θ_a for two perturbations $\theta_c = 0.1, 0.3$, respectively. We vary the step size and perturbation size to investigate the sensitivity of STAR-SPSA. The leftmost graph in both figures show that STAR-SPSA significantly outperforms SPSA for all choices of a_n, in terms of both MSE of the estimates as well as the throughput. Similarly, STAR-SPSA performs better than RM in both metrics with the exception of the case $\theta_a = 15$, when RM results in a slightly lower MSE of $\mathbf{x}_N$ and a higher MSE of $TP(\mathbf{x}_N)$, but the results are fairly close. The standard errors are within ± 0.1. In both Figures 3.5a and 3.5b, the STAR-SPSA results do not appear to be very sensitive to the step size, especially not the throughput. In fact, combining the direct and indirect gradients often improves the performance, and sometimes the difference can be significant, as in the case $\theta_a = 50$, $c_n = 0.1(n+1)^{-1/4}$. Furthermore, the STAR-SPSA mean throughput is always greater than SPSA, and either greater than RM very close to it. In addition, the confidence bands are much tighter for STAR-SPSA for both $\theta_c = 0.1, 0.3$. From these results, we conclude that incorporating the SP gradient will either improve the finite time performance or in the worst case, it will not hurt the performance significantly.

Orthogonal Projection Algorithm

- Step 0. Input: $\mathbf{X}_n$, M, c_n
- Step 1. Set $\mathbf{X}_{n+1} = \mathbf{X}_n$.
- Step 2. For $i = 1, \ldots, d$, if $X_{n,i} < c_n$, then set $X_{n+1,i} = c_n$.

- Step 3. Let
  - $\mathbf{P}_0 = M/d * \mathbf{1}$
  - $\mathbf{N} = \mathbf{1}/\sqrt{d}$
  - $\mathbf{T} = \mathbf{X}_{n+1} - \mathbf{P}_0$
  - $t =< \mathbf{T}, \mathbf{N} >$, where $< \cdot, \cdot >$ denotes dot product
  - $\mathbf{X}_{n+1} = \mathbf{X}_{n+1} - t\mathbf{N}$
- Step 4. If $X_{n+1,i} \geq c_n$ for all i, return $\mathbf{X}_{n+1}$. Otherwise, return $\mathbf{X}_n$.

Random Perturbation $\boldsymbol{\Delta}_n$

- Step 0. Input: d
- Step 1. If d is odd, generate $r \sim Unif\{1, d\}$, and set $\Delta_{n,r} = 0$. If d is even, let $r = 0$.
- Step 2. Generate $r_i \sim Unif(0, 1)$ for $i = 1, \ldots, d$, $i \neq r$.

  Let $m = \{i | r_i = median(r_1, \ldots, r_{r-1}, r_{r+1}, \ldots, r_d)\ \}$.
- Step 3. For $i = 1, \ldots, d$, $i \neq r$, if $r_i < m$, set $\Delta_{n,i} = -1$ and $\Delta_{n,i} = 1$ otherwise.
- Step 4. Return $\boldsymbol{\Delta}_n$.

IPA Algorithm: Closed Jackson Queueing Network

- Step 0. Set $\mathbf{A} = \mathbf{0}$, where $\mathbf{0}$ is a dxd zero matrix, $n = 1$, and N = number of customers serviced.

- Step 1. Suppose the next customer to finish service is currently located at node i. Increment $A_{i,i}$ by the service time s_i.

- Step 2. Suppose the customer leaving node i moves to node j. If node j is empty upon arrival, set $\mathbf{A}_{:,j} = \mathbf{A}_{:,i}$.

- Step 3. Set $n = n + 1$. If $n < N$, go back to Step 1. If $n = N$, set T= current time, go to Step 4.

- Step 4. Calculate minimum and maximum values of each row and set to $\mathbf{A}^- = (A_1^-, \ldots, A_d^-)$ and $\mathbf{A}^+ = (A_1^+, \ldots, A_d^+)$, respectively.

- Step 5. Compute $\frac{S_k}{TP} \cdot \frac{\partial TP}{\partial S_k} = -\frac{A_k^-}{T - A_k^+ + A_k^-}$, where S_k is the mean service time of server k and $TP = N/(T - A_k^+ + A_k^-)$.

IPA Algorithm: Closed Jackson Queueing Network w/Equality Constraint

- Step 1. Apply IPA gradient method for the unconstrained closed Jackson queueing network for throughput.

- Step 2. Generate $r \sim Unif\{1, d\}$.

- Step 3. For $k = 1, \ldots, d$, set $\frac{\partial TP_c}{\partial X_k} = \frac{\partial TP}{\partial X_k} - \frac{\partial TP}{\partial X_r}$.

3.4 Summary and Future Work

We have introduced the first set of stochastic approximation algorithms that integrate both direct and indirect gradient estimates for single and multidimensional problems. Our new hybrid gradient estimates use a symmetric finite difference-type gradient estimate for the indirect gradient and an average of the associated direct gradients for the direct gradient. The crux of the algorithm lies in the convex weight derived to minimize the variance of the hybrid gradient estimates. Under mild conditions, we have analytically shown the improvement over individual direct and indirect gradient estimates in terms of variance, which in turn can improve the SA performance. We have demonstrated the promise of STAR-SA numerically on a one-dimensional stylized problem as well as STAR-SPSA on a multidimensional queueing network. Previous work had only considered SA with either direct or indirect gradients, but we exploit the information contained in the indirect gradient estimates, which we have shown to be theoretically and practically beneficial.

Our hybrid technique can be generalized to include other gradient estimates within the two general gradient categories, e.g., IPA with LR/SF. In addition, we can adaptively choose both the weight sequence and the perturbation sequence.

Currently, the variance minimizing weights (3.2) for the homogeneous case are deterministic and depend on both the gradient and function noise, which are unknown. As $c_n \to 0$, the influence of the finite difference gradient estimate diminishes and the weight on the Tangents AveRaged gradient estimate approaches 1. Ideally, independent of the noise level, the weights will put more emphasis on

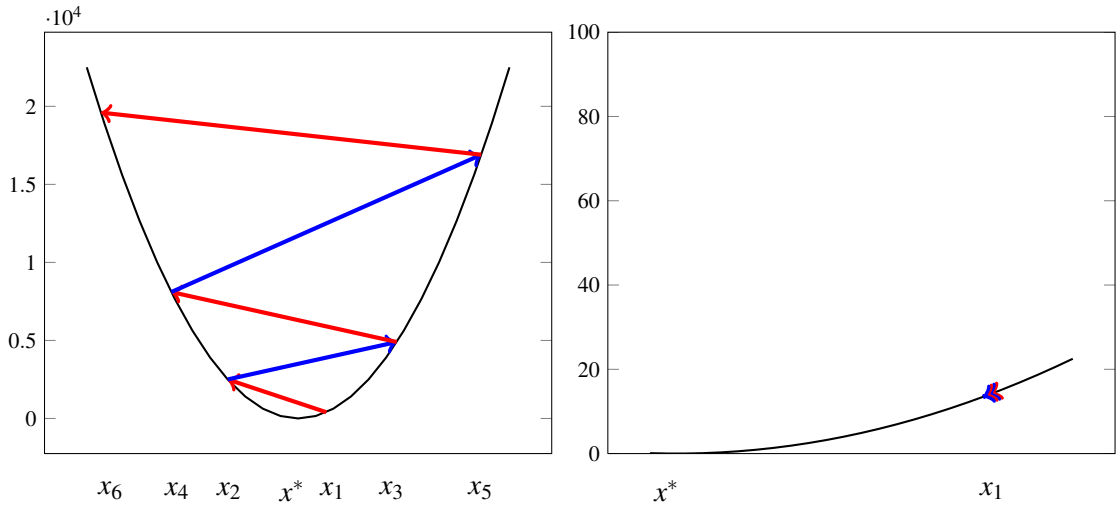
the gradient with higher accuracy. Since we are in the setting where both direct and indirect gradients available, we have access to various gradient approximations, i.e., $f'(x_n \pm c_n)$, $\frac{f'(x_n+c_n)+f'(x_n-c_n)}{2}$, and $\frac{f(x_n+c_n)-f(x_n-c_n)}{2c_n}$. At each iteration, these values can be used to determine which gradient is more accurate, g_S or g_{TAR}. If the sample performances are noisy and the perturbation size is very small, then the finite difference gradient could be very inaccurate, so ideally, less emphasis is placed on g_S, but c_n should not decrease since small perturbation sizes can lead to noisy gradients. Instead, we propose to decrease α_n in two ways: 1) decrease c_n and 2) increase the constant $(\sigma_f/\sigma_g)^2$. We can explore how to make the adjustments.

Chapter 4

Step Size Selection in Stochastic Approximation

4.1 Sensitivity of Finite-time Performance to Step Size

The asymptotic theory of stochastic approximation algorithms guarantees almost sure convergence under certain conditions. The most common requirements restrict the step size sequence $\{a_n\}$, but even a smaller subset of step size options still allow for an uncountable number of choices. It is impossible to find a universally optimal deterministic step size unless more information is known about the geometric structure of the function. Given a step size, one can always find a function where it performs poorly. For example in the one-dimensional case, if we consider a large step size applied to a very steep function, to ensure the iterates have an opportunity to move around the feasible region, then the iterates might oscillate back and forth, moving further and further away from the optimum for an extended period before making any progress towards the true solution, as seen in Figure 4.1a. The opposite could also occur, where the function is extremely flat and the step size is relatively small in comparison, so as the number of iterations increases, the iterates barely make any progress, and once the stopping time is reached, the algorithm returns a poor estimate in the region where it is currently "stuck," illustrated in Figure 4.1b. The finite-time performance of stochastic approximation algorithms is sensitive to the step size, and there are clear disadvantages for using an arbitrarily



(a) a_n is "too large" relative to the gradient (b) a_n is "too small" relative to the gradient

Figure 4.1: Illustration of Sensitivity of SA to $\{a_n\}$.

chosen deterministic step size. To circumvent this issue, adaptive step sizes have been proposed to adjust based on the ongoing performance of the algorithm. Ideally, an adaptive step size algorithm is able to recognize/detect the current estimates' proximity to the true solution, path behavior/trend, and geometric structure near the current iterate, while adjusting the step size accordingly. Before we propose our new adaptive step sizes, we conduct preliminary tests on existing finite-time theory on MSE bounds and two adaptive step sizes, Kesten's rule and SSKW.

4.1.1 KW and its Variants

For our preliminary numerical experiments, we focus on the one-dimensional KW algorithm, which generates $\widehat{\nabla} f(X_n)$ in (1.2) using finite differences. Although theoretical convergence can be guaranteed by satisfying certain requirements, practical performance depends on the choice of tuning sequences. In addition to selecting

a gain sequence $\{a_n\}$ in (1.2), the KW algorithm requires an additional task of choosing a finite difference perturbation sequence $\{c_n\}$ for the gradient. The finite-time performance of KW depends on both sequences $\{a_n\}$ and $\{c_n\}$. Because of the sensitivity of the KW algorithm to the tuning sequences, it is essential to choose an appropriate pair. In practice, KW could have the following shortcomings: long oscillatory period if the gain sequence $\{a_n\}$ is "too large," degraded convergence rate if $\{a_n\}$ is "too small," and poor gradient estimates if the perturbation sequence $\{c_n\}$ is "too small."

We conduct an empirical investigation of the sensitivity of KW and two of its adaptive variants, namely Kesten's rule and the scaled-and-shifted KW (or SSKW) algorithm of [7]. Our goal is to identify problem characteristics that exert a strong impact on algorithm performance, even in the presence of theoretical guarantees. For example, in the numerical results reported in [7], SSKW outperforms the KW algorithm in terms of both MSE and oscillatory behavior in finite time; however, this result is obtained using what seem to be nearly worst-case parameter settings for KW. We replicate these results, but we also find that the performance of KW can be significantly improved over a fairly wide choice of parameter settings. Although the worst-case performance of SSKW is much better than that of KW, it is also the case that KW provides the best performance in a significant proportion of problem instances. In addition, we find that Kesten's rule performs similar to KW, and sometimes better, when both algorithms begin with the same initial start value. We also investigate the finite-time MSE bound in [7] and characterize instances where this bound is tight. These results underscore the well-known difficulty of tuning,

even for adaptive versions of KW.

4.2 Finite-time MSE Bound

Asymptotic convergence properties of the KW algorithm and its variations have been a major research focus in SA. Convergence proofs in MSE appear in [18], [16], [19], [58], and [49] for various assumptions and modifications of the KW algorithm. However, in practice, where the run-time is finite, a good finite-time bound for the MSE is useful.

By applying similar technique as in [18], [7] derived a finite-time bound for the MSE of the KW algorithm. The MSE bound depends on certain problem-dependent constants, which are typically difficult to calculate in practice. However, in the special case where f is quadratic, the bound can be computed in closed form, allowing us to observe its tightness. We briefly summarize the bound as follows. First, we make the following assumptions on the function $f(x)$:

1. There exist positive constants K_0, K_1, and C_0 such that for every $c \in [0, C_0]$,

$$-K_1(x - x^*)^2 \leq \frac{f(x+c) - f(x-c)}{c}(x - x^*) \leq -K_0(x - x^*)^2.$$

2. $f'(x)(x - x^*) < 0$ for all $x \in R \backslash \{x^*\}$.

We also assume that the tuning sequences satisfy:

1. $a_n/c_n^2 \leq (a_{n+1}/c_{n+1}^2)(1 + Aa_{n+1})$ for all $n \geq 1$,

2. $a_n \to 0$ as $n \to \infty$,

with $0 < A < 2K_0$. Then,

$$E(X_{n+1} - x^*)^2 \leq Ca_n/c_n^2 \text{ for all } n \geq 1, \tag{4.1}$$

where C is a constant explicitly defined as

$$C = \max\left\{\frac{\sigma^2}{\xi}, \max_{1\leq n\leq n_0}\left\{\frac{c_n^2}{a_n}B_{n+1}\right\}\right\},$$

and

$$\begin{aligned}
D_n &= K_1^2 A a_n^2 + (K_1^2 - 2AK_0)a_n - 2K_0 - A, \\
n_0 &= 1 \text{ if } D_n < 0 \text{ for all } n \geq 1 \text{ and} \\
&\quad \sup\left\{n \geq 1 : (K_1^2 - 2AK_0)a_n + K_1^2 A a_n^2 \geq 2K_0 - A\right\} + 1 \text{ otherwise}, \\
\xi &= -\sup\{A - 2K_0 + (K_1^2 - 2AK_0)a_n + K_1^2 A a_n^2 : n \geq n_0\}, \\
B_n &= X_1^2 \prod_{i=1}^{n} p_i + \sum_{i=2}^{n-1} q_i \prod_{j=i+1}^{n} p_j + q_n, \\
p_i &= 1 - 2a_i K_0 + K_1^2 a_i^2, \text{ for } i = 1, 2, \ldots, n, \\
q_i &= \frac{a_i^2}{c_i^2}\sigma^2, \text{ for } i = 1, 2, \ldots, n, \\
\sigma^2 &= \sup_{x\in\Theta} \mathrm{Var}[\tilde{f}(X_n + c_n) - \tilde{f}(X_n - c_n)|X_n = x].
\end{aligned}$$

Equation (4.1) does not guarantee convergence in MSE, but rather establishes a finite bound for each iteration. The bound is thus more useful when it is tight. In Section 4.3.1, we investigate the tightness of (4.1) by comparing the bound with the exact MSE of simple quadratic functions of the form $f(x) = \alpha x^2$ where $\alpha < 0$ and the optimal $x^* = 0$. The exact MSE can be computed as follows:

$$E(X_{n+1} - x^*)^2 = X_1^2 \Pi_{i=1}^{n}(1 + 2\alpha a_i)^2 + \frac{\sigma^2}{2}\sum_{k=1}^{n-1}\frac{a_k^2}{c_k^2}\Pi_{j=k+1}^{n}(1 + 2\alpha a_j)^2 + \frac{a_n^2\sigma^2}{2c_n^2}. \tag{4.2}$$

4.3 Numerical Experiments

4.3.1 Tightness of the Finite-time MSE Bound for Quadratics

We generated the MSE bound in (4.1) and the exact MSE in (4.2) for quadratic functions with various noise levels and initial starting values for three different cases: 1) $f(x) = -0.001x^2$, $c_n = 1/n^{1/2}$, $f(x) = -0.15x^2$, $c_n = 1/n^{1/4}$ and $f(x) = -0.15x^2$, $c_n = 1/n^{1/2}$. The MSE bound is a function of constants that are not unique, satisfying S1, S3, A1, and A2. We picked the largest K_0 and smallest K_1 satisfying A1 and A slightly less than $2K_0$. Table 4.2 lists the constants used in our calculations for the MSE bound in (4.1), and the exact MSE and MSE bound are listed in Table 4.1. The exact MSE (4.2) is a sum of three components. The first term on the right hand side (RHS) of (4.2) is independent of σ and is dominated by the initial starting value, X_1. The second and third terms in (4.2) are dominated by σ. When $\sigma \in \{0.001, 0.01, 0.1, 1.0\}$, both terms are small ($< 1$), but when $\sigma = 10$, the RHS is dominated by the second term. Therefore, the exact MSE increases with X_1 and σ. Using the parameters in Table 5.1, the constant C in the MSE bound can be expressed as

$$C = \max\left\{\frac{\sigma^2}{\xi}, \frac{c_1^2}{a_1}\left(X_1(1-2a_1K_0+K_1^2a_1^2)(1-2a_2K_0+K_1^2a_2^2)+\frac{a_2^2}{c_2^2}\sigma^2\right)\right\}. \quad (4.3)$$

The first term in (4.3) dominates when σ is large since $\xi = 0.001, 0.1$ for $f(x) = -0.001x^2, -0.15x^2$, respectively. Therefore, the MSE bound and difference between the exact MSE and MSE bound increases significantly when σ increases from 1.0 to 10.0. Otherwise, the MSE bound is equal to the second term, which

Table 4.1: Finite-time MSE bound and exact MSE for KW with $n = 10000,\ a_n = 1/n$

		$f(x) = -0.001x^2$, $c_n = 1/n^{1/2}$		$f(x) = -0.15x^2$, $c_n = 1/n^{1/4}$		$f(x) = -0.15x^2$ $c_n = 1/n^{1/2}$	
σ	X_1	Exact	Bound	Exact	Bound	Exact	Bound
0.001	0	0.00	0.00	0.00	0.00	0.00	0.00
	-5	24.04	24.85	0.06	8.85	0.06	0.09
	-10	96.16	99.40	0.24	35.40	0.24	0.35
	-20	384.64	397.61	0.95	141.61	0.95	1.42
	-40	1538.56	1590.42	3.78	566.44	3.78	5.66
0.01	0	0.00	0.10	0.00	0.00	0.00	0.00
	-5	24.04	24.85	0.06	8.85	0.06	0.09
	-10	96.16	99.40	0.24	35.40	0.24	0.35
	-20	384.64	397.61	0.95	141.61	0.95	1.42
	-40	1538.56	1590.42	3.78	566.44	3.78	5.66
0.1	0	0.05	10.0	0.01	0.10	0.00	0.00
	-5	24.09	24.86	0.07	8.86	0.06	0.09
	-10	96.21	99.41	0.24	35.41	0.24	0.35
	-20	384.69	397.61	0.95	141.62	0.95	1.42
	-40	1538.61	1590.43	3.79	566.45	3.78	5.66
1.0	0	4.8	999.99	0.83	10.00	0.03	0.10
	-5	28.84	999.99	0.89	10.00	0.09	0.10
	-10	100.96	999.99	1.07	35.90	0.27	0.36
	-20	389.44	999.99	1.78	142.11	0.98	1.42
	-40	1543.36	1590.92	4.61	566.94	3.81	5.67
10.0	0	480.08	99999.20	83.23	999.79	3.20	9.99
	-5	504.12	99999.20	83.29	999.79	3.26	9.99
	-10	576.24	99999.20	83.47	999.79	3.43	9.99
	-20	864.72	99999.20	84.18	999.79	4.14	9.99
	-40	2018.64	99999.20	87.01	999.79	6.98	9.99

Table 4.2: Finite-time MSE Bound Parameters for KW

$f(x)$	a_n	c_n	K_0	K_1	A	n_0	ξ
$-0.001x^2$	$1/n$	$1/n^{1/2}$	0.002	0002	0.003	1	0.001
$-0.15x^2$	$1/n$	$1/n^{1/2}$	0.3	0.3	0.5	1	0.1
$-0.15x^2$	$1/n$	$1/n^{1/4}$	0.3	0.3	0.5	1	0.1

increases with X_1 and σ. Table 4.1 contains the exact MSE and MSE bound for three different parameter settings and for $\sigma \in \{0.001, 0.01, 0.1, 1.0, 10.0\}$. The first column in Table 4.1 presents results for $f(x) = -0.001x^2,\ c_n = 1/n^{1/2}$. In the presence of more noise, i.e. $\sigma = 10.0$, the MSE bound is $99,999.20$, which is the first term in (4.3) for each initial starting value. The difference between this bound and the exact MSE is significant with a difference greater than $97,500$. For $\sigma = 1.0$, the MSE bound only takes the second term in (4.3), when the starting position is farther from the optimum, i.e. $X_1 = -40$ and is tight. However, when the initial starting value is closer to the optimum, i.e. $X_1 = 0, -5, -10, -20$, the MSE bound is equal to 999.99, which is the first term in (4.3), and thus the MSE bound is significantly greater than the exact MSE. The MSE bound is very tight for rest of the cases with the exception of when $\sigma = 0.1$ and $X_1 = 0$. For the second column with $f(x) = -0.15x^2,\ c_n = 1/n^{1/2}$, the MSE bound is significantly greater than the exact MSE across the board. The third column reports results for $f(x) = -0.15x^2, c_n = 1/n^{1/2}$ the MSE bound is tight for all cases with the exception of the case with $\sigma = 10.0$. It would seem that the bound is a useful guideline for problems with low variance, but becomes less tight as the noise level increases.

4.3.2 Sensitivity of KW and its Variants

We compare the MSE performance between KW and two of its variants described in Section 2.2.3, Kesten's rule and SSKW. All experiments were implemented with $a_n = \theta_a/n,\ c_n = \theta_c/n^s$ where $s \in \{1/4, 1/2\},\ \theta_a > 0,\ \theta_c > 0$, 10000 iterations, and 1000 sample paths.

Not surprisingly, the performance of SSKW relative to KW heavily depends on the chosen parameters such as truncated interval length, initial starting value, and tuning sequences. Our analysis replicates the results of [7], where SSKW performs significantly better than KW in terms of MSE and oscillatory period, but we find that the chosen parameters for this experiment are among the worst possible parameters for KW as illustrated in Figure 4.2 with KW and SSKW under $\theta_a = \theta_c = 1$. By choosing a different initial starting position, the performance of KW can be significantly improved, as demonstrated in Table 4.3 for two functions $f(x) = -0.001x^2$ and $f(x) = 100e^{-0.006x^2}$. To offer a contrast with the quadratic function, the second function considered is very steep and has flat tails.

[7] compared the SSKW performance with that of KW whose MSE is highly reliant on the tuning sequences and initial start value. The MSE performance results for $f(x) = -0.001x^2$ using KW in [7] were poor because the initial position was chosen to be far from the optimum and the gain size a_n was too small to make any noticeable progress towards it after 10000 iterations, so the iterates hover around the initial position. In our numerical experiments, we also consider $a_n = \frac{\theta_a}{n}$ and $c_n = \frac{\theta_c}{n^{1/4}}$ for $\theta_a, \theta_c > 0$. If $\theta_a = \theta_c = 1$ as in [7], but the initial start value is 0.01

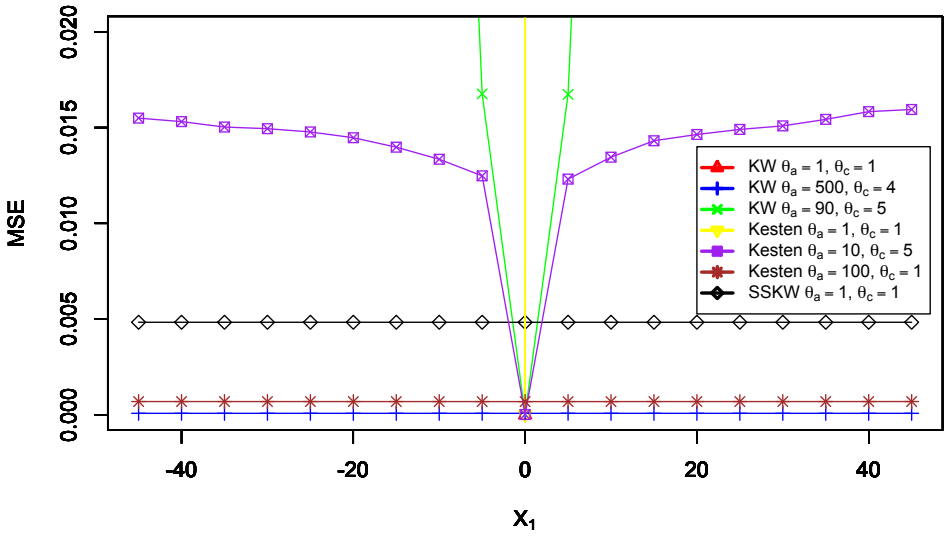


Figure 4.2: MSE of the 10000th iterate of KW and Kesten for three parameter settings and SSKW for $f(x) = -0.001x^2$, $\sigma = 0.001$, $a_n = \theta_a/n$, $c_n = \theta_c/n^{1/4}$.

instead of 30, then the MSE from KW is significantly lower compared to SSKW. The first column in Table 4.3 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.

We also conduct a sensitivity analysis for $f(x) = -0.001x^2$ with various starting positions X_1 and multiplicative constants, θ_a, and θ_c and implement SSKW and KW using Kesten's rule. For the sensitivity analysis, we considered a wide selection of parameters: 19 initial starting values uniformly spaced within the truncated

		$f(x) = -0.001x^2$ [-50, 50] $X_1 = 0.01$			$f(x) = 100e^{-0.006x^2}$ [-50, 50] $X_1 = 30$		
σ	Alg.	100	1000	10000	100	1000	10000
.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
	Kesten	$1.12x10^{-4}$	$1.08x10^{-4}$	$1.04x10^{-4}$	10^{-7}	$3x10^{-8}$	10^{-8}
.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
	Kesten	$2.10x10^{-3}$	$2.11x10^{-3}$	$2.05x10^{-3}$	$9.54x10^{-6}$	$2.76x10^{-6}$	$8.41x10^{-7}$
.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
	Kesten	$2.01x10^{-1}$	$2.03x10^{-1}$	$1.97x10^{-1}$	$5.65x10^{-2}$	$2.76x10^{-4}$	$8.41x10^{-5}$
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
	Kesten	20.1	20.3	19.7	456.9	315.1	239.7

Table 4.3: MSE of the 100th, 1000th, and 10000th iteration for KW and its variates with $a_n = 1/n,\ c_n = 1/n^{1/4}$.

interval $X_1 \in \{-50 + 5k \mid k = 1, 2, ...19\}$, 45 different θ_a values parametrized by $\theta_a \in \{10^s k \mid k = 1, 2, \ldots, 9, s = 0, 1, \ldots, 4\}$ and 10 different θ_c values parametrized by $\theta_c \in \{10^s k \mid k = 1, 2, ..., 5, s = 0, 1\}$. In total, there are 8550 possible combinations of parameters.

The results show that KW and Kesten's rule are sensitive to the parameter choice, but near-optimal performance can be obtained with tuning. Figure 4.2 plots the MSE of KW for $f(x) = -0.001x^2, \sigma = 0.001$ against the initial starting values X_1 for different sets of parameter choices. These cases serve as a good representation of the majority of the MSE behaviors among the entire set of results. The case with $\theta_a = \theta_c = 1$ is among the worst for KW and Kesten's rule. The MSE is represented by a nearly vertical line for both algorithms. For this parameter setting, SSKW beats

KW and Kesten's rule significantly for all initial values with the exception of $X_1 = 0$. For the case where $\theta_a = 90, \theta_c = 5$, KW outperforms SSKW in a neighborhood around the optimum. However, there are cases such as $\theta_a = 500,\ \theta_c = 4$ for KW and $\theta_a = 100,\ \theta_c = 1$ for Kesten's rule that outperform SSKW for all initial start values. Of the 8550 combinations varying all parameters and 450 combinations with $X_1 = 30$, KW performs better than SSKW in 4275 and 215 cases, respectively, suggesting that KW requires some tuning to perform well, but that there is a fairly wide range of tunable parameters that yield good performance. If KW performs better than KW, the difference is not as pronounced as when SSKW outperforms KW, but careful tuning can partially mitigate KW's sensitivity to parameters such as the initial iterate.

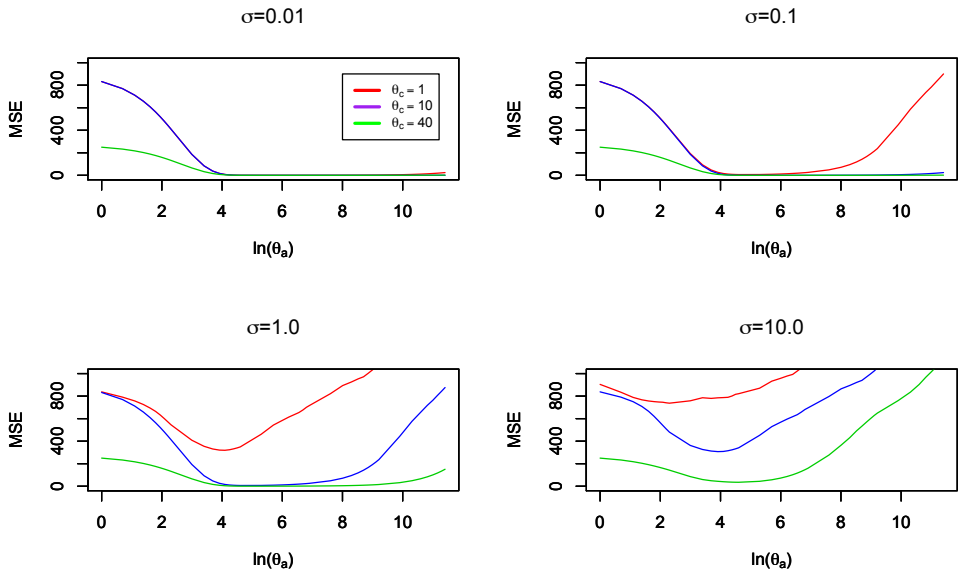


Figure 4.3: Sensitivity of KW to θ_a for $f(x) = -0.001x^2$, $a_n = \theta_a/n,\ c_n = \theta_c/n^{1/4},\ n = 10000$.

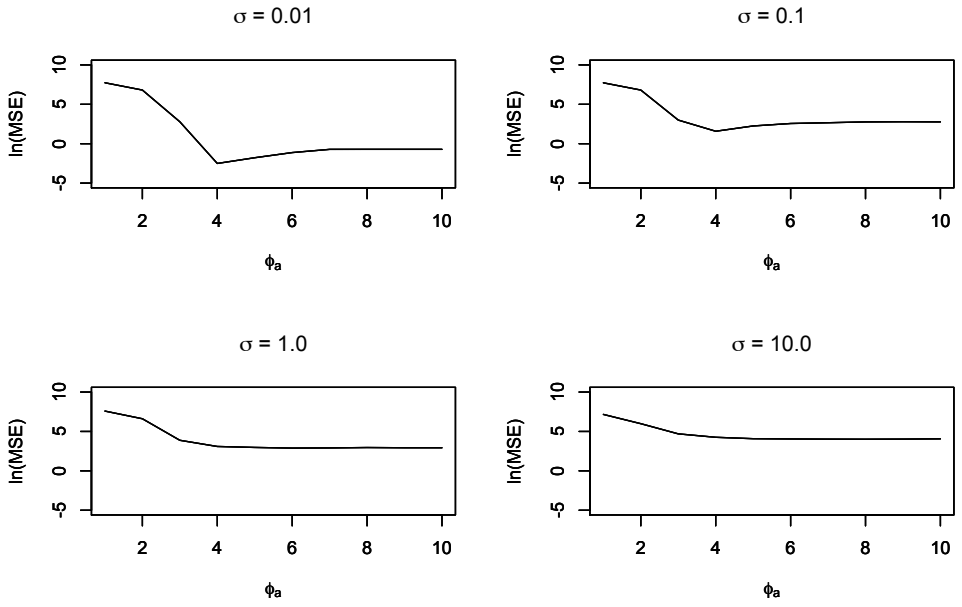


Figure 4.4: Sensitivity of SSKW for $f(x) = -0.001x^2$ as a function of $\phi_a,\ n = 10000$.

Figure 4.3 plots the MSE $f(x) = -0.001x^2, a_n = \theta_a/n, c_n = \theta_c/n^{1/4}$of the 10000th iterate as a function of $\log\theta_a$ given $\theta_c = 1, 10, 40$. The case where $\sigma = 0.001$ is omitted, because the results are similar to those for $\sigma = 0.01$. For $\log\theta_a < 4$, the MSE decreases for each given value of θ_c. However, for $\log\theta_a \geq 4$, the MSE behaves differently for all noise levels. But, the overall behavior as a function of θ_c is similar across noise levels. The MSE decreases for all θ_a as θ_c increases, so in the case where $\theta_c = 40$, there is a wide range of θ_a values where the MSE of KW is lower than that of SSKW. But the MSE of KW could also be extremely high if the tuning sequences are not chosen well. Moreover, we investigate the sensitivity of SSKW to ϕ_a, which is the upper bound of the scale up factor for $\{a_n\}$ as depicted in Figure 4.4. The MSE decreases until ϕ_a, the maximum scale up factor for $\{a_n\}$, is equal to

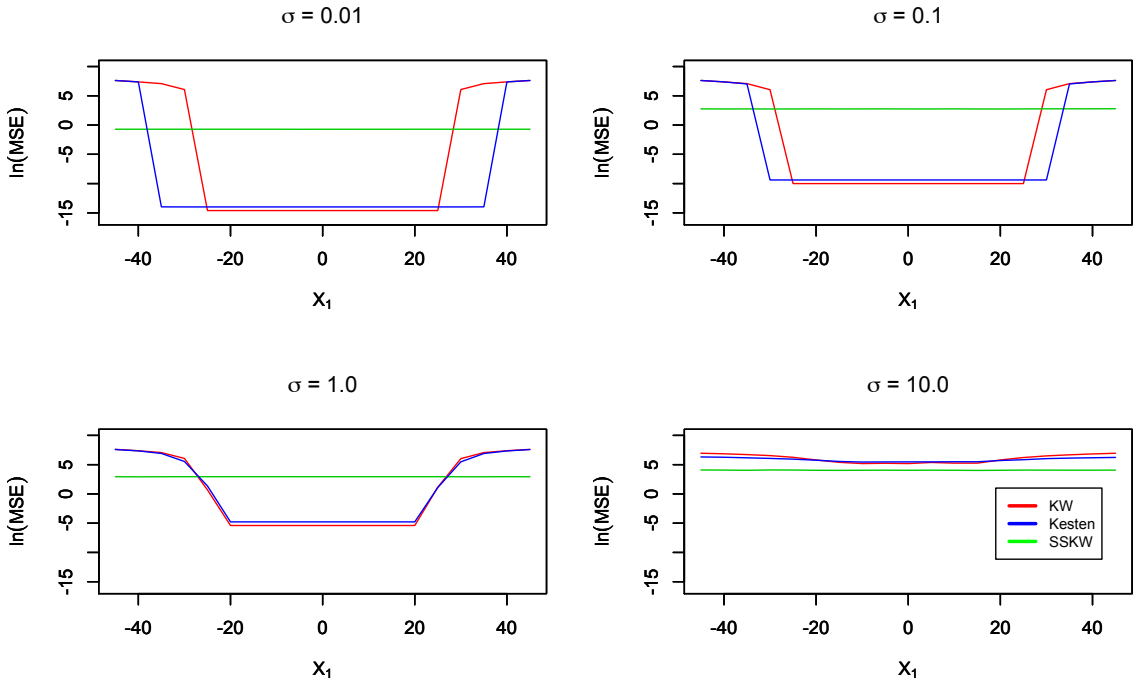


Figure 4.5: MSE Comparison of KW, Kesten, and SSKW for $f(x) = 100e^{-0.006x^2}$, $a_n = 1/n$, $c_n = 1/n^{1/4}$, $n = 10000$.

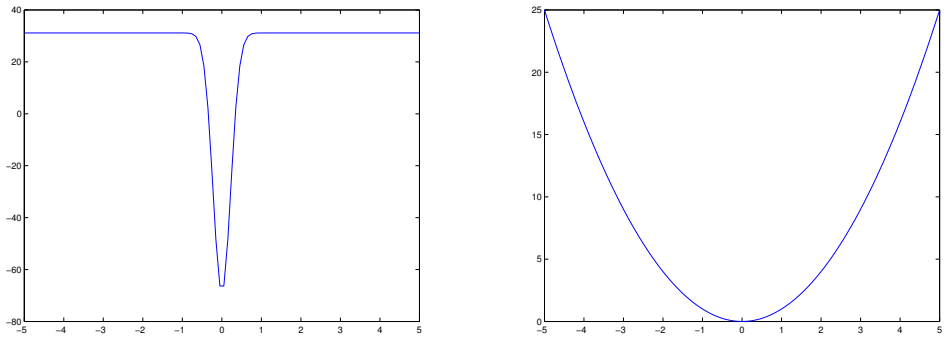
4 and increases for $\sigma = 0.01, 0.1$ while it levels off for $\sigma \in \{1.0, 10.0\}$ thereafter. It seems that for lower noise levels,i.e. $\sigma \in \{0.01, 0.1\}$, $\phi_a = 4$ is a better choice, while $\phi_a = 10$ leads to a lower MSE for $\sigma \in \{1.0, 10.0\}$.

In addition, we implement KW and its variants using the same parameters (i.e., $a_n = 1/n, c_n = 1/n^{1/4}, X_1 = 30$) as in [7] on $f(x) = 100e^{-0.006x^2}$ to test the algorithms under the same setting for a different function. Figure 4.5 plots the MSE of the 10000th iterate as a function of the initial start value. KW and Kesten's rule outperform SSKW within certain intervals around the optimum for $\sigma \in \{0.001, 0.01, 0.1, 1.0\}$ and Kesten's better performance intervals overlap the intervals of KW. However, the KW using the deterministic step-size $1/n$ performs

better than using Kesten's step-size where the intervals overlap, which can be seen in Figure 4.5. Unfortunately, outside of those intervals, both algorithms have a tendency to perform poorly. However, for the other four noise levels, the intervals where KW and Kesten's rule outperform SSKW are larger. However, there is a tradeoff, since if by chance the initial start value is closer to the boundary, the difference in performance can be drastic.

4.4 PROX-step

The idea behind our adaptive step size can be easily conveyed in the two examples illustrated in Figures 4.6a and 4.6b. For a well-behaved function without drastic changes in its gradient in neighborhoods, as depicted in Figure 4.6b, there is a possibility of selecting a step size that leads to good finite-time performance, where the iterates can approach the minimum at a good pace without heavy oscillations. However, the function in Figure 4.6a has a steep valley region around the optimum, whereas the rest of the function is extremely flat. This type of function arises in the likelihood function typically used to find maximum likelihood estimators. If we use a single step size sequence, the performance will often be poor since the two extreme regions require two drastically different step sizes. For instance, a larger step size is appropriate in the flat area to avoid getting stuck at a poor estimate; however, once the iterate reaches the valley region, the larger step size is no longer appropriate, and the recursion will overshoot and launch the estimate back into a flat region. This cycle will continue until the step size is sufficiently small so that



(a) Contrasting regions **(b)** Well-behaved

Figure 4.6: Illustration of PROX-Step Motivational

the iterates make progress towards the minimum. This is only possible if the step size was small before entering the valley region, which means many iterations were necessary to reach that point.

We propose an adaptive step size, inspired by both Kesten's rule and SSKW, that adjusts based on sample performances, gradient estimates, and parameter estimates, taking advantage of past data. Kesten's rule is Markovian-like, since the next step size depends on the previous gradient (or estimate) as well as the current gradient (or estimate). On the other hand, SSKW's adaptive mechanisms depend largely on the feasible search space/region and the two most recent parameter estimates. While the aforementioned adaptive step sizes have significantly advanced SA algorithms, they have their pitfalls. Kesten's rule starts with an arbitrary non-increasing step size, which can still fall into shortcomings as in the deterministic case if a_n is too large. It does however, prevent the step size from decreasing if a_n is too small (assuming consecutive gradients are the same sign). An extension by [53]

allows the step size to increase, but it can be slow. SSKW certainly addresses the issue of implementing SA with an inappropriately small or large a_n, but different regions in the feasible search space call for different step sizes. For instance, if the current iterate is in a very steep portion of the function, then under SSKW, it will reduce the step size faster. However in the next iteration after the step size reduction, the estimate could be located in a flat region, which requires a larger a_n to accelerate the convergence to a neighborhood of the optimum. In the worst case, the estimate gets stuck in a region because of the newly reduced a_n is too small, since aside from the initial phase, SSKW can only reduce a_n and cannot enlarge/increase it.

Although the underlying notion of our adaptive step size is motivated by Kesten and SSKW, we approach adaptivity from a different angle. Our method adopts a memory-based approach, where the step size is adjusted based on the perceived proximity to the optimum as in Kesten and is increased/decreased when deemed necessary as in SSKW, but the adjustments are determined by the *sample performance(s)* in addition to the gradient(s) and parameter estimate(s) used in other methods. Each simulation run used to obtain the sample performance and gradient estimate (if available) is expensive and the values generated contain information that can be used in future iterations to improve the SA performance. We efficiently exploit the data collected in previous iterations to aid in the selection of an appropriate step size to be used in the current update.

For illustrative purposes, consider the simple one-dimensional KW algorithm. At each iteration, two sample performances are generated to compute the finite

difference gradient estimate for the recursive update. After collecting a sufficient number of observations, we can roughly guess how close the current estimate is from the optimum based on the moving average of sample performances in comparison to the current observation. If the current iterate is considered close, then a smaller step size could prevent the next iterate from completely overshooting. In contrast, if the estimate is believed to be further away, then a larger step size could decrease the amount of effort required to put the iterate in closer range of the optimum. For a minimization problem, the estimate is perceived to be closer to the optimum if the current objective value is less than the mean of the past objective values, and further from the optimum if it is greater than the mean, i.e.,

- $f_n > \bar{f}_n$ (farther),
- $f_n < \bar{f}_n$ (closer),

where f_n is the average of sample performances generated at the nth iteration and $\bar{f}_n$ is the running average of all sample performances (i.e., for KW with symmetric differences, $f_n^+ = f(x_n+c_n)$, $f_n^- = f(x_n-c_n)$, $f_n = (f_n^+ + f_n^-)/2$, and $\bar{f}_n = \frac{1}{n}\sum_{i=1}^{n} f_i$. Our adaptive step size begins with two positive fixed sequences $\{a_n^+\}$ and $\{a_n^-\}$, where $a_n^- < a_n^+$ for all n. We first tailor the adaptive step size to SA algorithms that involve a symmetric finite difference gradient (e.g., KW or STAR-SA), and then generalize it. We now present PROX-step for one-dimensional SA algorithms involving a symmetric finite difference, followed by a generalization.

PROX-step for KW or STAR-SA

- Step 0. Input: $x_1 \in \Theta$, $\{a_n^+\}$, $\{a_n^-\}$, $\{c_n\}$, N.
- Step 1. Initialize $\bar{f}_0 = 0$, $n = 1$.
- Step 2. Generate f_n^+ and f_n^- to compute $\widehat{\nabla} f(x_n)$.
- Step 3. Update mean objective value $\bar{f}_n = \frac{n-1}{n}\bar{f}_{n-1} + \frac{1}{n} f_n$.
- Step 4. Set
$$a_n = \begin{cases} a_n^- & \text{if} \quad f_n < \bar{f}_n, \\ a_n^+ & \text{if} \quad f_n > \bar{f}_n. \end{cases}$$
- Step 5. Update $x_{n+1} = \Pi_\Theta \left(x_n - a_n \widehat{\nabla} f(x_n) \right)$.
- Step 6. If $n < N$, set $n = n + 1$, go to Step 2.
- Step 7. Output: $x_N^* = x_N$

Generalized PROX-step for SA

- Step 0. Input: $\mathbf{x}_1 \in \Theta$, $\{a_n^+\}$, $\{a_n^-\}$, N.
- Step 1. Initialize $\bar{f}_0 = 0$, $n = 1$.
- Step 2. Generate n_d sample performances $f_n^{(i)}$ for $i = 1, \ldots, n_d$ to compute $\widehat{\nabla} f(\mathbf{x}_n)$.

- Step 3. Update mean objective value $\bar{f}_n = \frac{n-1}{n}\bar{f}_{n-1} + \frac{1}{n}f_n$, where $f_n = \frac{1}{n_d}\sum_{i=1}^{n_d} f_n^{(i)}$.

- Step 4. Set
$$a_n = \begin{cases} a_n^- & \text{if} \quad f_n < \bar{f}_n, \\ a_n^+ & \text{if} \quad f_n > \bar{f}_n. \end{cases}$$

- Step 5. Update $\mathbf{x}_{n+1} = \Pi_\Theta\Big(\mathbf{x}_n - a_n\widehat{\nabla} f(\mathbf{x}_n)\Big)$.

- Step 6. If $n < N$, set $n = n + 1$, go to Step 2.

- Step 7. Output: $\mathbf{x}_N^* = \mathbf{x}_N$

4.5 Adaptive PROX-step

Although the PROX-step is adaptive, the two sequences used to generate it are both deterministic, so in the end, it still faces some of the same shortcomings of implementing a deterministic step size. Each sequence $\{a_n^+\}$ and $\{a_n^-\}$ could be inappropriately selected for its corresponding feasible region; therefore, we propose a method to adjust the sequences based on past events. In particular, throughout the recursion, in addition to the necessary information used to implement the PROX-step, we track the step size used at each iteration to determine whether or not the smaller sequence $\{a_n^-\}$ appropriate. If the step sizes used are alternating between $\{a_n^+\}$ and $\{a_n^-\}$ consistently, then it is an indication that either we are close to the optimum or the smaller step size is too big. In either case, it would be ideal to

decrease the smaller step size sequence. Furthermore, in applying Kesten's rule, we do not have a separate step sizes for each dimension, so instead, we look at the the total number of sign changes in the gradient and if more than half of the dimensions keep the same sign, then we keep the same two step size sequences for the next iteration (i.e., $\{a_n^{\pm}\} = \{a_{n-1}^{\pm}\}$). Lastly, we look at the magnitude of each violation $||x_{n+1} - \Pi(x_{n+1})||^2$ and of each iteration change $||x_n - \Pi(x_{n+1})||^2$ and reduce or increase the step sir sequence accordingly while maintaining $a_n^- < a_n^+$ still holds. Step size adjustments can be done in the following ways:

- Scaling
- Kesten's rule

Scaling will be used for decreasing and increasing the step size by dividing and multiplying by the scaling factor, respectively, and Kesten's rule will keep the step size constant.

Additional input parameters must be chosen:

- $\alpha > 1$ is the scale factor
- $\{d_n\}$ is the projection violation threshold
- $\{e_n\}$ is the no-movement threshold
- n_L is the number of allowable scales

The asymptotic theory still applies, since the scaling and shifting will only occur finitely many times.

Adaptive PROX-step

- Step 0. Input: $\mathbf{x}_1 \in \Theta$, $\{a_n^+\}$, $\{a_n^-\}$, $\{d_n\}$, α, n_L, N.
- Step 1. Initialize $\bar{f}_0 = 0$, $n = 1$, $n_L = 0$.
- Step 2. Generate n_d sample performances $f_n^{(i)}$ for $i = 1, \ldots, n_d$ to compute $\widehat{\nabla} f(\mathbf{x}_n)$.
- Step 3. Update mean objective value $\bar{f}_n = \frac{n-1}{n}\bar{f}_{n-1} + \frac{1}{n} f_n$.
- Step 4. Set

$$a_n = \begin{cases} a_n^- & \text{if} \quad f_n < \bar{f}_n, \\ a_n^+ & \text{if} \quad f_n > \bar{f}_n. \end{cases}$$

- Step 5. Set

$$a[n] = \begin{cases} -1 & \text{if} \quad f_n < \bar{f}_n, \\ 1 & \text{if} \quad f_n > \bar{f}_n. \end{cases}$$

- Step 6. Update $x_{n+1} = \Pi_\Theta \left(x_n - a_n \widehat{\nabla} f(x_n) \right)$.
- Step 7. Modify step size sequences.
  - **Scaling down.**
    - If $n_S < n_L$ and $||\mathbf{x}_{n+1} - \Pi(\mathbf{x}_{n+1})||^2 > d_n$, and
      - if $a_n = a_n^+$, then set $\{a_n^+\} = \{a_n^+/\alpha\}$, $n_S = n_S + 1$.
      - if $a_n = a_n^-$, then set $\{a_n^-\} = \{a_n^-/\alpha\}$, $n_S = n_S + 1$.

* If $n > 3$, $n_S < n_L$, $a[n]a[n-1] < 0$, $a[n-1]a[n-2] < 0$, and $a[n-2]a[n-3] < 0$, then set $\{a_n^-\} = \{a_n^-/\alpha\}$ and $n_S = n_S + 1$.

– **Scaling up.**

* If $n_S < n_L$ and $0 < ||\mathbf{x}_n - \Pi(\mathbf{x}_{n+1})||^2 < e_n$, and

· if $a_n = a_n^+$, then set $\{a_n^+\} = \{\alpha a_n^+\}$, $n_S = n_S + 1$.

· if $a_n = a_n^-$, then set $\{a_n^-\} = \{\alpha a_n^-\}$, $n_S = n_S + 1$.

– **Kesten.**

* If $\sum_{i=1}^{d} I_{\{\widehat{\nabla} f_i(\mathbf{x}_n) \cdot \widehat{\nabla} f_i(\mathbf{x}_{n+1}) > 0\}} > d/2$, then $\{a_n^+\} = \{a_{n-1}^+\}$ and $\{a_n^-\} = \{a_{n-1}^-\}$.

- Step 8. If $n < N$, set $n = n + 1$, go to Step 2.

- Step 9. Output: $\mathbf{x}_N^* = \mathbf{x}_N$

Unfortunately, the input parameters must be chosen by the users, which can have a significant effect on the performance. In our numerical experiments, we choose the parameters conservatively so that the original two sequences do not change dramatically from one iteration to the next.

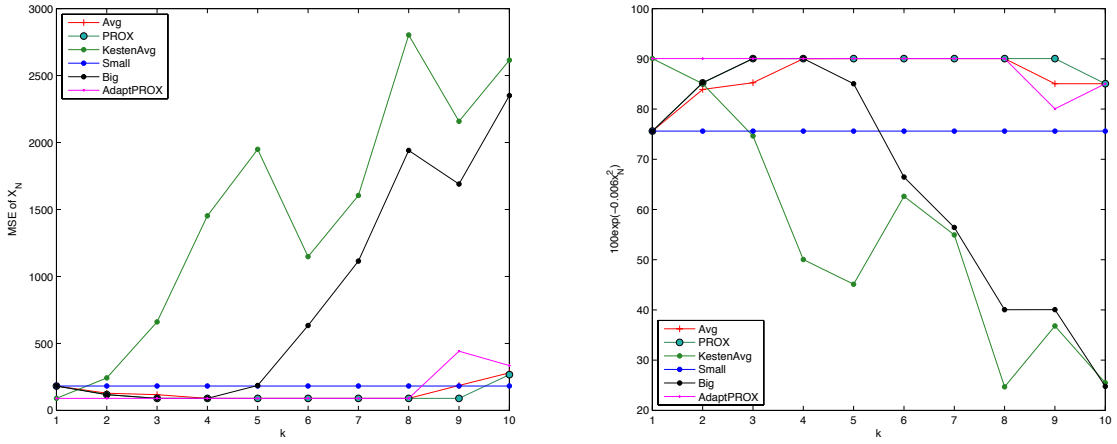
4.6 Numerical Experiments

We investigate the performance of PROX-step and aPROX-step on two contrasting functions with added noise as well as the queueing network example described in Section 3.4.

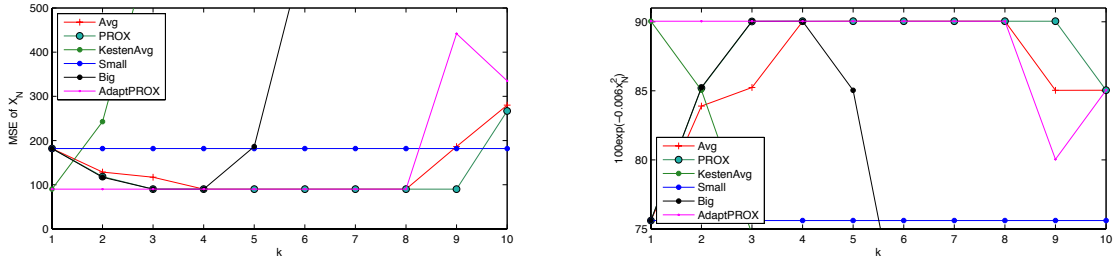
4.6.1 Deterministic Problem with Added Noise

For the one-dimensional case, we consider two deterministic functions: 1) exponential that is very steep near the optimum with very flat tails and 2) a simple quadratic. We compare the standard decreasing step size, Kesten's rule, and the PROX-step and Adaptive PROX-step applied to RM. For the fixed decreasing step sizes, we look at three different step size parameters (i.e., from the PROX-step, Big: $\{a_n^+\}$, Small: $\{a_n^-\}$, Average: $\{(a_n^+ + a_n^-)/2\}$). We apply Kesten's rule to the average step size sequence $\{(a_n^+ + a_n^-)/2\}$ and use PROX and aPROX with the parameter $d_n = 10$ for all n. The initial value $x_1 = 10$ and feasible region $\Theta = [-30, 70]$ were arbitrarily chosen. We consider $N = 50$ to be the stopping time, 20 macro replications, and two step size settings for $a_n^{\pm}$.

To demonstrate the robustness of PROX and aPROX, we fix the smaller sequence $\{a_n^-\}$, vary the larger sequence $\{a_n^+\}$, and plot the MSE of the estimate x_N and $f(x_N)$ for both $f(x) = 100e^{-0.006x^2}$ and $f(x) = -x^2$. Figure 4.7 presents results for the exponential case with $a_n^- = 10/n$ and $a_n^+ = 5(k+1)/n$, where k is the x-axis, where Figure 4.7a is the full plot and Figure 4.7b is a magnified version. Clearly, Kesten's rule applied to the average step size $\{(a_n^+ + a_n^-)/2\}$ and the larger step size $\{a_n^+\}$ perform poorly as a_n^+ increases, since their MSEs of the estimate x_N are much higher than the rest in the left figure, and the function value $f(x_N)$ is much lower in the right graph (higher is better since we are maximizing). The performance of using the fixed small step size $\{a_n^-\}$ is represented by the horizontal lines since it is independent of a_n^+, performs better than Big and KestenAvg, but both PROX

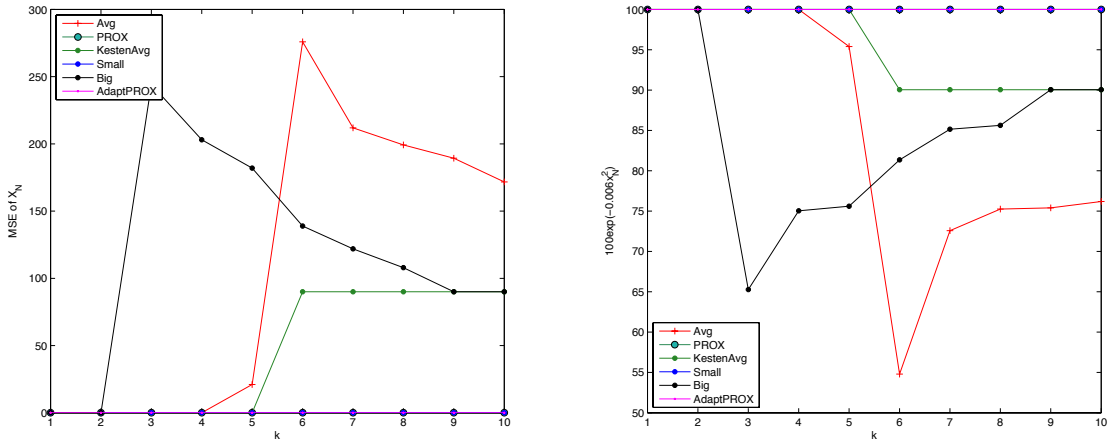


(a) MSE of x_N and $f(x_N)$

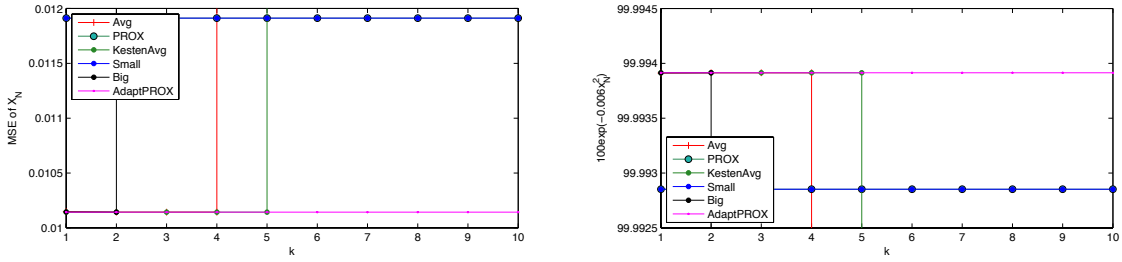


(b) Magnified version

Figure 4.7: $f(x) = 100e^{-0.006x^2}$, $\theta = [-30, 70]$, $x_1 = 10$, $a_n^- = 10/n$, $a_n^+ = 5(k+1)/n$, $\sigma_f = \sigma_g = 0.1$, $\alpha = 1.025$, $n_L = 20$, $N = 50$, macroreplications $= 20$.

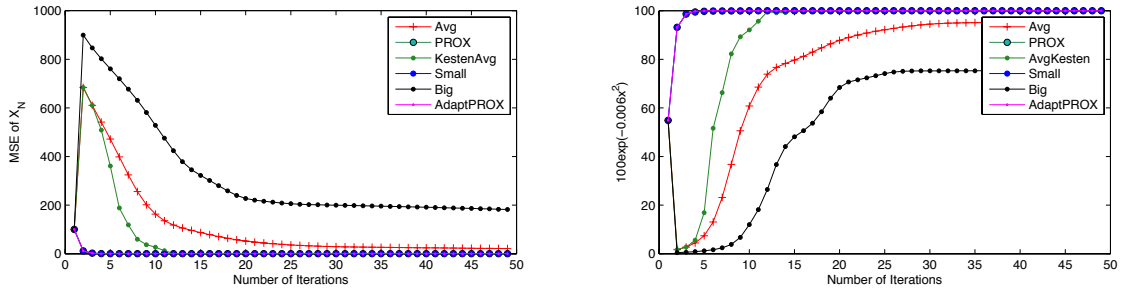


(a) MSE of x_N and $f(x_N)$

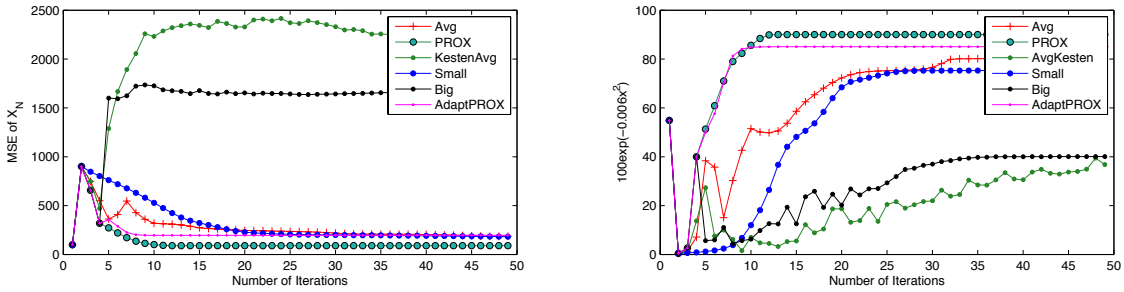


(b) Magnified version

Figure 4.8: $f(x) = 100e^{-0.006x^2}$, $\theta = [-30, 70]$, $x_1 = 10$, $a_n^- = 1/n$, $a_n^+ = 2k/n$, $\sigma_f = \sigma_g = 0.1$, $\alpha = 1.025$, $n_L = 20$, $N = 50$, macroreplications = 20.



(a) $a_n^- = 1/n$, $a_n^+ = 10/n$

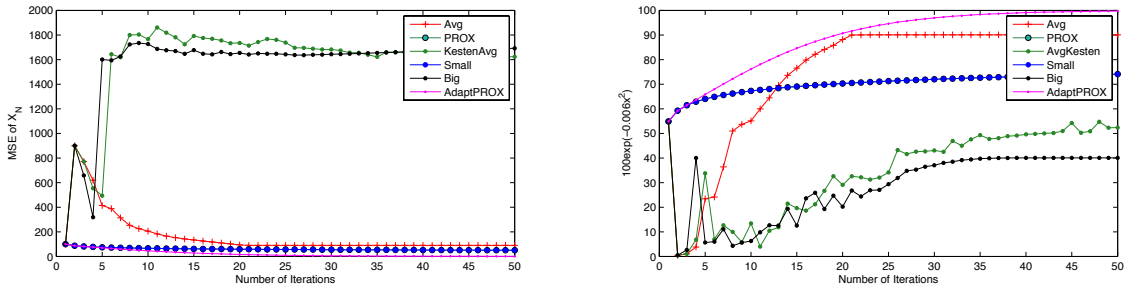


(b) $a_n^- = 10/n$, $a_n^+ = 50/n$

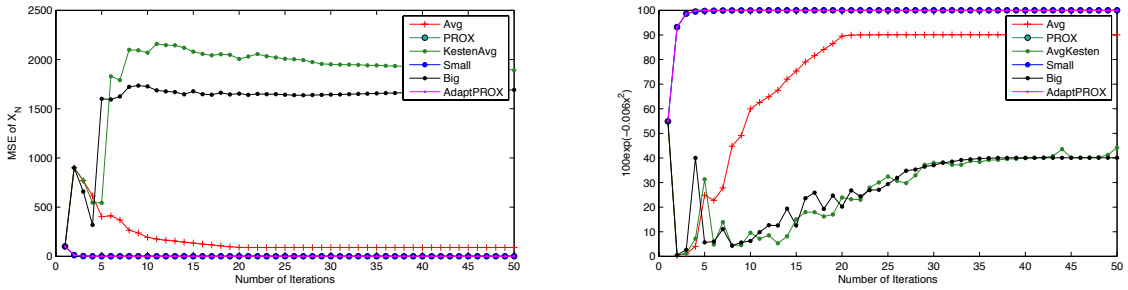
Figure 4.9: MSE of x_N and $f(x_N)$, $f(x) = 100\exp^{-0.006x^2}$, $\alpha = 1.5$, $\theta = [-30, 70]$, $x_1 = 10$, $\sigma_f = \sigma_g = 0.1$,, $\alpha = 1.5$, $n_L = 50$, $N = 50$, macroreplications $= 20$.

and aPROX appear to outperform in almost all cases. Figure 4.7b zooms in on the better performances for a clearer comparison. Figure 4.8 presents results for a similar setting with the exception of step sizes, where, $a_n^- = 1/n$ and $a_n^+ = 2k/n$. The reduction in step sizes result in the same two step sizes (Big and KestenAvg) performing poorly, but the average step size sequence also joins the group. From Figure 4.8b, aPROX outperforms all algorithms for the range of step sizes $\{a_n^+\}$, and the PROX results are not far behind.

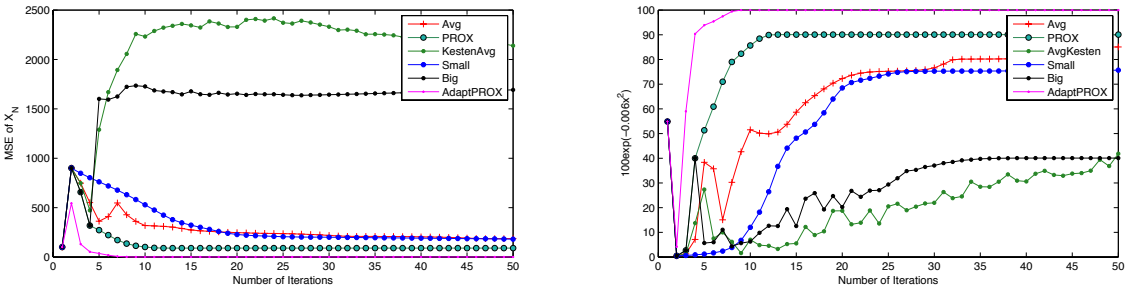
We also plot the MSE of x_N and the function $f(x_N)$ as a function of the



(a) $a_n^- = .1/n$, $a_n^+ = 50/n$

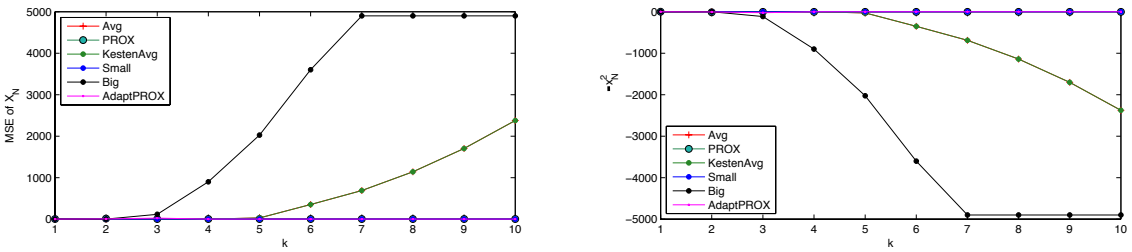


(b) $a_n^- = 1/n$, $a_n^+ = 50/n$

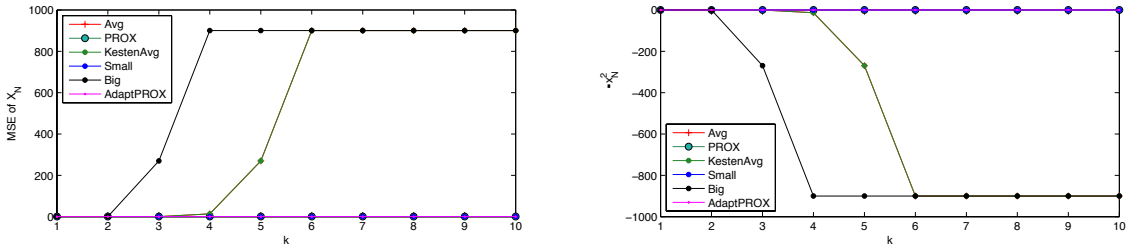


(c) $a_n^- = 10/n$, $a_n^+ = 50/n$

Figure 4.10: MSE of x_N and $f(x_N)$, $f(x) = 100\exp^{-0.006x^2}$, $\alpha = 2$ $\theta = [-30, 70]$, $x_1 = 10$, $\sigma_f = \sigma_g = 0.1$, $n_L = 20$, $N = 50$, macroreplications $= 20$.



(a) $N = 10$, $a_n^- = 1/n$, $a_n^+ = 2k/n$



(b) $N = 50$, $a_n^- = 10/n$, $a_n^+ = 5(k+1)/n$

Figure 4.11: MSE of x_N and $f(x_N)$, $f(x) = -x^2$, $\theta = [-30, 70]$, $x_1 = 10$, $\sigma_f = 0.1$, $\sigma_g = 0.1$, $n_L = 20$, macroreplications $= 20$.

iteration in Figure 4.9 for two different step size settings. Figure 4.9a and 4.9a illustrate the results for the exponential function with smaller and larger step sizes, respectively. For the smaller step sizes, (i.e., $a_n^- = 1/n$ and $a_n^+ = 10/n$), all of the algorithms with the exception of Big seem to be approaching the optimum, but PROX, aPROX, and Small do so earlier on, followed by KestenAvg and then Avg. With a larger step size, KestenAvg and Big perform very poorly, while the others make progress as the number of iterations increase. From Figure 4.9b, PROX appears to outperform all algorithms followed by aPROX. Moreover, Figure 4.10 also plots the MSE of x_N and $f(x_N)$ for a similar setting, but we increase the scale factor α from 1.5 to 2. For all three subfigures $a_n^+ = 50/n$, but the smaller step size a_n^- increases from 0.1/n to 1/n to 10/n. In Figures 4.10a and 4.10c, aPROX performs

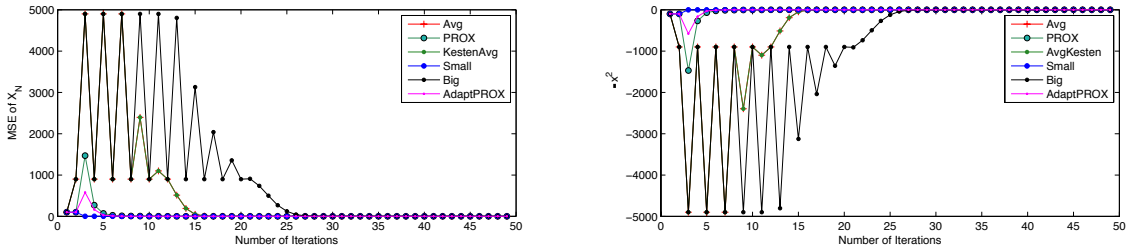


Figure 4.12: MSE of x_N and $f(x_N)$, $f(x) = -x^2$, $N = 50$, $x_1 = 10$, $\sigma_f = 0.1$, $\sigma_g = 1.0$, $n_L = 50$, $a_n^- = 1/n$, $a_n^+ = 20/n$

the best followed by PROX, and both algorithms outperform the rest. These figures show that PROX has promise and aPROX can improve the performance of PROX in certain cases. The performance for PROX and aPROX in Figure 4.10b are almost identical.

We also test the adaptive step sizes on the simple quadratic function $f(x) = -x^2$ for the same two step size settings. The results are similar to those in the exponential case, which can be seen in Figures 4.11a, 4.11b, and 4.12. This results shows that PROX and aPROX can be applied to both well-behaved and ill-behaved functions. In all of the cases, PROX and aPROX perform similarly, which is not a surprise since we used conservative parameter settings.

4.6.2 9-station Closed Jackson Queueing Network

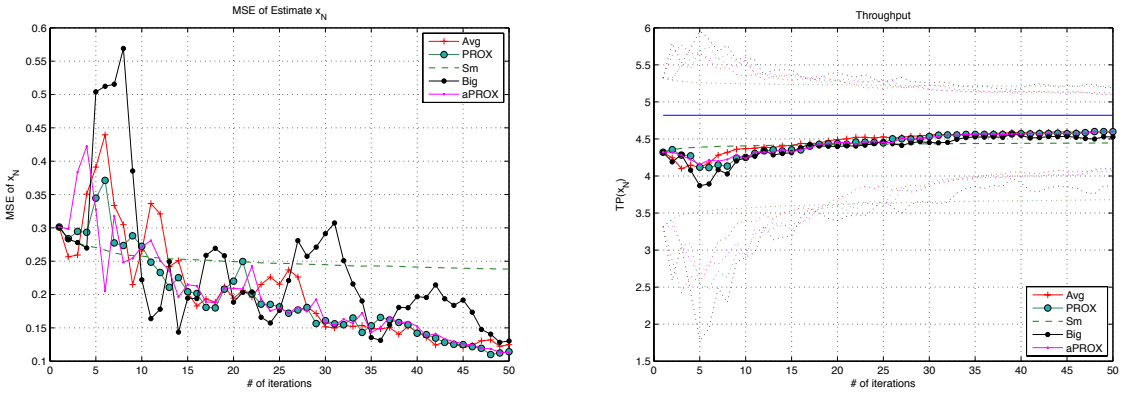
We empirically test the PROX-step and the aPROX-step on the close queueing network problem from Section 3.4 with identical settings with the exception of the step size parameter. Figures 4.13a, 4.13b, 4.13c, 4.14a, 4.14b, and 4.14c illustrate the results of the MSE of the $\mathbf{x}_N$ as well as the throughput $TP(\mathbf{x}_N)$ for various step

sizes for six different step size parameters. In all of the cases, PROX and aPROX are either in the top three algorithms or the top two at the stopping time. Figure 4.14c depicts an extreme case where the step sizes are either extremely small or large, but outperform the other algorithms.

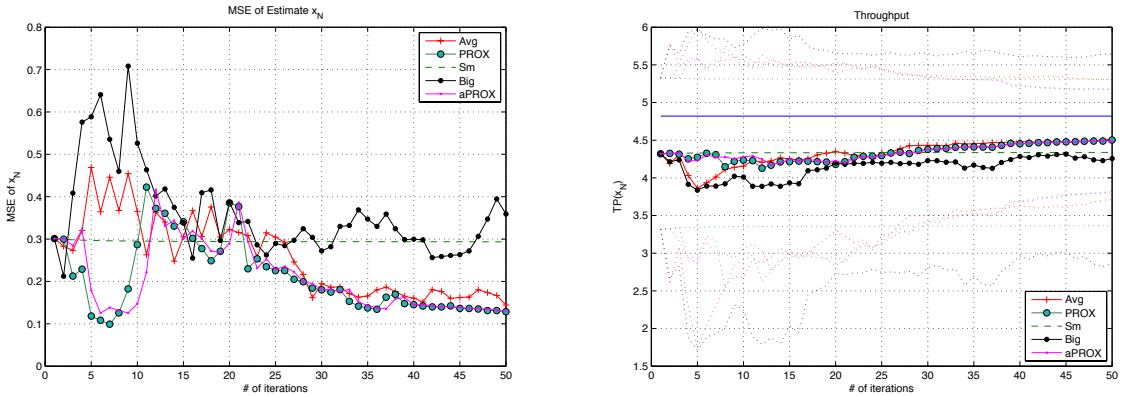
4.7 Summary and Future Work

We investigated the sensitivity of SA algorithms to step size sequences as well as the tightness of a finite-time MSE bound. In addition, we introduced the first adaptive step size algorithm based on two deterministic sequences from which the current step size is selected from, which is determined by the current and past sample performances. We also propose an adaptive method to adjust the two deterministic sequences based on current and past events. We empirically show the promise on stylized problems and a queueing network.

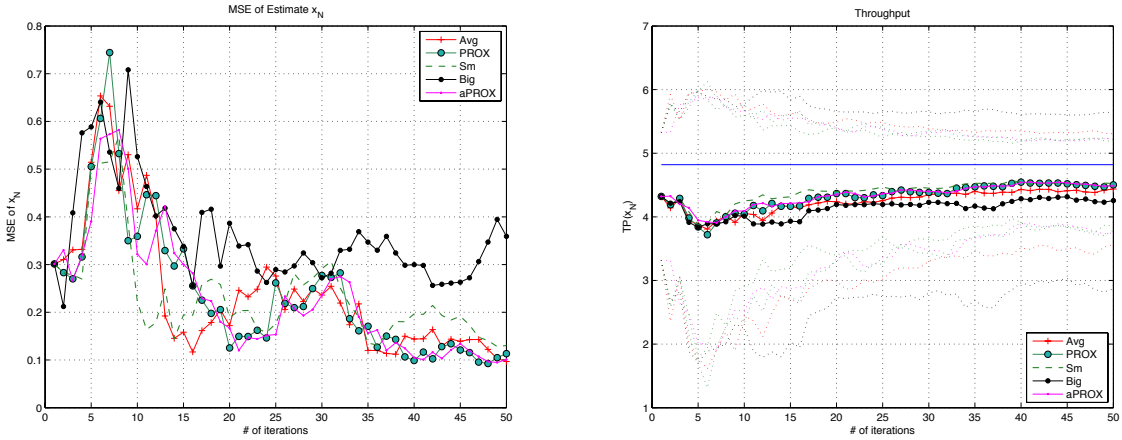
Unfortunately, the adaptive method in modifying the two initial sequences rely on user chosen parameters, so in the future, we would like to find ways to automatically select these parameters. In addition, we could apply this step size to other algorithms and numerical examples.



(a) $a_n^- = 0.1/n$, $a_n^+ = 10/n$, $\alpha = 1.5$, $d_n = 5$

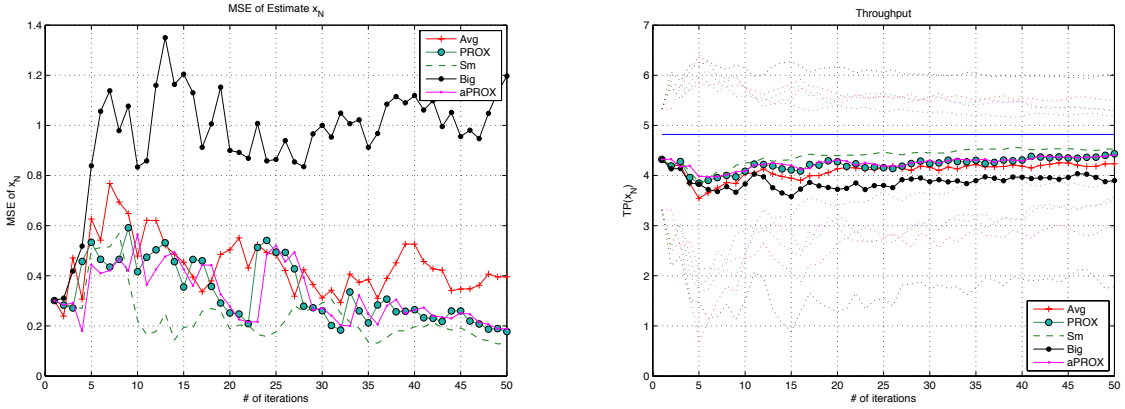


(b) $a_n^- = 0.01/n$, $a_n^+ = 20/n$, $\alpha = 1.5$, $d_n = 5$

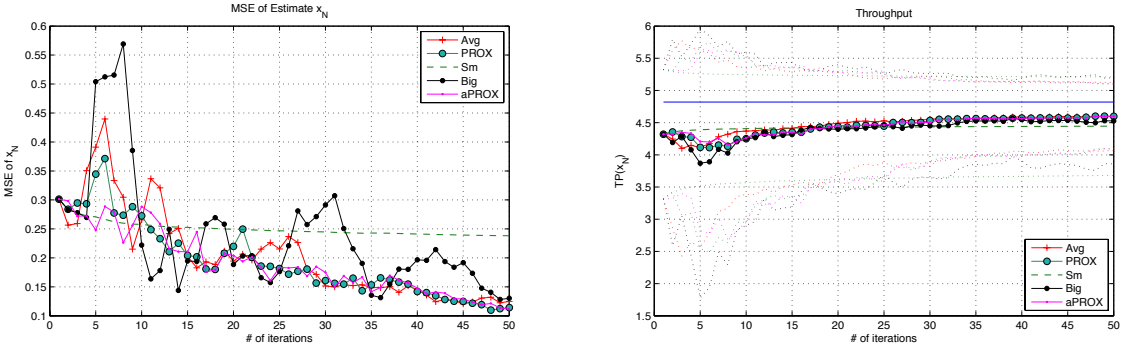


(c) $a_n^- = 10/n$, $a_n^+ = 20/n$, $\alpha = 1.5$, $d_n = 5$

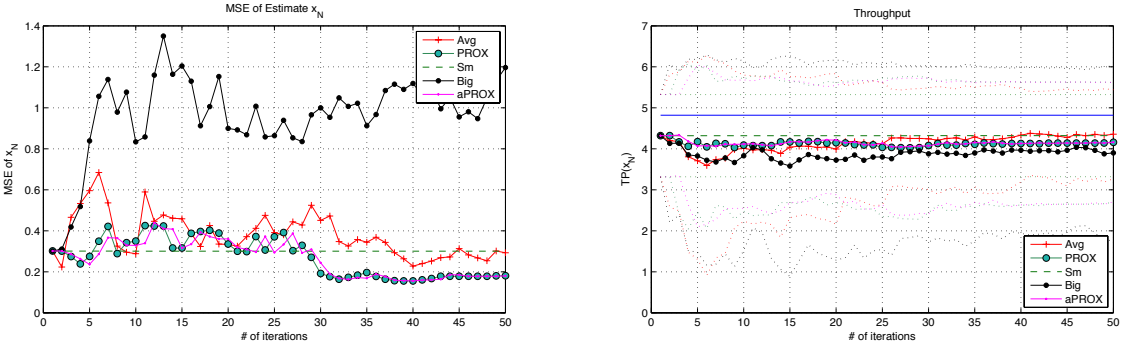
Figure 4.13



(a) $a_n^- = 10/n$, $a_n^+ = 50/n$, $\alpha = 1.5$, $d_n = 5$



(b) $a_n^- = 0.1/n$, $a_n^+ = 10/n$, $\alpha = 2$, $n_L = 20$



(c) $a_n^- = 0.001/n$, $a_n^+ = 50/n$, $\alpha = 2$, $n_L = 20$

Figure 4.14

Chapter 5

Greek Kernel Estimators

5.1 Introduction

Financial derivatives have become an integral part of risk management. Therefore, accessing and quantifying the risk of these assets have sparked great interest in the finance industry. The Greeks are derivatives of (financial) derivative prices with respect to an underlying market parameter. Each Greek measures a different dimension of market risk. For instance, "delta," "vega," and "theta" are first-order derivatives of the expected payoff with respect to price, volatility, and time, respectively, and "gamma" is the second-order derivative with respect to price.

One type of financial derivative used for hedging is an option, which gives the security holder the right to exercise the option once certain conditions are satisfied, which is dependent on the option type. For example, the exercise condition for an Asian digital option depends on depends on the average price over a pre-specified period, and it pays a lump sum if the option is "in the money" and zero otherwise. Up-and-out barrier call options can be exercised once the price reaches the strike price but does not cross the upper barrier. Both of these options have discontinuous payoff functions, and the option price can be written as

$$E[g(S) \cdot I_{\{h(S)\geq 0\}}], \tag{5.1}$$

where the expectation is taken under the risk-neutral measure, S is the price modeled by a stochastic process, $I_{\{\cdot\}}$ is the indicator function, and $g(S)$ is the discounted payoff when the exercise condition, $h(S) \geq 0$, is satisfied. The price S depends on the market parameters $\theta \in \Theta$, where Θ is an open set. For simplicity, assume θ is one-dimensional, since we can focus on one market parameter and hold all else constant. The derivative of the payoff function (5.1) with respect to θ is a first-order Greek, and the derivative of the first-order Greek with respect to θ is a second-order Greek, both of which we wish to estimate. Some well-known derivative estimation methods include finite difference approximation, perturbation analysis (also known as the pathwise method), the score function or likelihood ratio (SF/LR) method, and the weak derivative (WD) method. Each method has its advantages and drawbacks.

The most straightforward approach to estimate Greeks is the finite difference method. Although this method is easy to implement, the resulting estimators are biased and generally have a high mean-squared error (MSE) [33]. In addition, a perturbation sequence $\{\Delta\theta\}$ must be selected, which affects the bias/variance tradeoff, although symmetric differences can be used to reduce the order of bias [24]. In addition, smaller perturbations could lead to more noise because of the stochasticity. The pathwise method, generally known as infinitesimal perturbation analysis (IPA) in the simulation community, is easy to implement and often performs well in comparison with other methods, but unfortunately, it is not always applicable. For instance, it is not suitable for discontinuous functions nor second-order Greeks [43]. Smoothed perturbation analysis (SPA) is able to overcome the discontinuity issue, and [28] suggest using SPA to smooth out the discontinuity by conditioning on ap-

propriate random variables (e.g., [61]). The key, which is the most difficult part of this method, is choosing the random variable on which to condition. On the contrary, the weak derivative (WD) method is applicable to discontinuous functions, but may require a high number of simulations [24]. The likelihood ratio method can also approximate the gradient of discontinuous functions, but usually leads to high variance. For a comprehensive review of gradient estimation methods, refer to [33], [23], and [4].

[43] introduces a modified pathwise method to estimate first- and second-order Greeks for options with discontinuous payoffs. It is a generalization of the classical pathwise method, which extends to options with discontinuous payoffs (5.1) by rewriting the Greek as a sum of an expectation and a derivative(s) with respect to an auxiliary parameter. The general pathwise method and kernel method are applied to the expectation and derivative terms, respectively. The accuracy of the kernel estimator depends/hinges on the chosen kernel function, which relies on a bandwidth/smoothing parameter, described in more detail in Section 5.3.3. A pilot simulation proposed in [43] generates an "optimal" bandwidth parameter prior to applying the modified pathwise method. The pilot simulation involves various input parameters, which affect the bandwidth output. The modified pathwise method has been applied to estimate delta, vega, theta and gamma of the Asian digital option and up-and-out barrier call option.

In this chapter, we examine the accuracy of the Greeks estimated using a modified pathwise gradient estimation method for an Asian digital option and up-and-out barrier call option by conducting two sets of numerical experiments. The first set

investigates the sensitivity of the Greek estimators to the bandwidth parameter(s), and the second explores the sensitivity of a proposed method used to generate "optimal" bandwidths for the modified pathwise method to various input parameters. Our numerical results show that the Greek estimators are quite sensitive to the bandwidth, so the performance of the pilot simulation is critical.

5.2 Problem Setting

We focus on options with a discontinuous payoff function that can be written in the form in (5.1), where $g(\cdot)$ and $h(\cdot)$ are differentiable functions, and S is a vector of discretized prices. Let S_{t_i} denote the price of the security at time $t_i \geq 0$, where $t_i = iT/k$ for $i = 0, 1, 2, \ldots, k$, which are evenly spaced time points between 0 and T. For simplicity, denote S_{t_i} by S_i, and $S = (S_0, S_1, \ldots, S_k)'$. Let $p(\theta) = E[f(S)]$, where $p'(\theta) = \partial p(\theta)/\partial\theta$ is the first-order Greek with respect to θ and $p''(\theta) = \partial^2 p(\theta)/\partial\theta^2$ is the second-order Greek with respect to θ. If $\theta = S_0$, $\theta = \sigma$, and $\theta = \tau$, then $p'(\theta)$ is called delta, vega, and theta, respectively, and if $\theta = S_0$, then $p''(\theta)$ is called gamma.

We explore Asian digital options and up-and-out barrier call options, both of which have discontinuous payoffs. For the Asian digital option, the discounted payoff function $g(S) = e^{-rT}$, where r is the risk-free rate and T is the duration of time considered, and the exercise criterion is based on an average of k discretized prices $\overline{S} = \frac{1}{k}\sum_{i=1}^{k} S_i$ exceeding a pre-specified threshold K over a predetermined period of time, i.e., whether or not $h(S) \triangleq \overline{S} - K \geq 0$. The up-and-out barrier call

Table 5.1: Parameters.

	r	σ	b	μ	S_0	K	T	U	k
Asian digital option	0.05	0.30	0.2	98	100	100	1	-	10, 20, 50
up-and-out barrier call option	0.05	0.20	-	-	100	100	1	120	20, 50

option can be exercised if the final price S_k is greater than or equal to a threshold K, but the maximum price $S_{max} = \max\{S_1, \ldots, S_k\}$ never exceeds an upper barrier U, i.e., whether or not $h(S) = \min\{S_k - K, U - S_{max}\} \geq 0$. The payoff function is the amount by which the final price S_k exceeds the strike price K discounted by e^{-rT}, i.e., $g(S) = e^{-rT}(S_k - K)$.

In the numerical experiments, the prices considered for the Asian digital option and the up-and-out barrier call option follow an Ornstein-Uhlenbeck process and a geometric Brownian motion generated, respectively, using

$$\begin{aligned} S_i &= S_{i-1}e^{-b\tau} + \mu(1 - e^{-b\tau}) + \sigma\sqrt{(1 - e^{-2b\tau})/(2b)}Z_i \quad \text{for } i = 1, 2, ..., k, \\ S_t &= S_0 e^{(r-\sigma^2/2)t+\sigma B_t} \quad \text{for } t = T/k, 2T/k, ..., T, \end{aligned}$$

where b, σ, μ, r, k, and τ are the mean reversion rate, volatility, mean return, risk-free interest rate, number of discretized intervals, and incremental time step (T/k), respectively. The variables Z_i and B_t are independent standard normal random variables and standard Brownian motion, respectively. The values considered are listed in Table 5.1.

5.3 Generalized Pathwise Method

The pathwise method is in general not applicable to second-order Greeks nor discontinuous functions, but the modified kernel version is one method that circumvents this issue.

5.3.1 First-Order Greeks

The first-order Greek estimator based on the modified pathwise method is based on the following theoretical result from [43].

Assumption 1. *For any $\theta \in \Theta$, $g(S)$ and $h(S)$ are differentiable with respect to θ with probability 1 (w.p.1), and there exist random variables K_g and K_h with finite second moments that may depend on θ, such that $|g(S(\theta+\Delta\theta))-g(S(\theta))| \leq K_g|\Delta\theta|$ and $|h(S(\theta+\Delta\theta))-h(S(\theta))| \leq K_h|\Delta\theta|$ when $|\Delta\theta|$ is sufficiently small.*

Assumption 2. *For any $\theta \in \Theta$, $\partial_\theta \phi(\theta, y)$ exists and is continuous at (θ, y) with $\phi(\theta, y) = E[g(S) \cdot 1_{\{h(S)\geq y\}}]$.*

Theorem 5.3.1. *Suppose $g(\cdot)$ and $h(\cdot)$ satisfy Assumptions 1 and 2, and $E[|g(S)|^2] < +\infty$ and $E[|h(S)|^2] < +\infty$. Then*

$$p^{'}(\theta) = E[\partial_\theta g(S) \cdot 1_{\{h(S)\geq 0\}}] - \partial_y E[g(S)\partial_\theta h(S) \cdot 1_{\{h(S)\geq y\}}]|_{y=0} \tag{5.2}$$

The two terms on the right hand side of (2) can be estimated easily, because the first term is an ordinary expectation, where $g(S)$ is differentiable with respect to θ, and the second term is a derivative with respect to y, which is independent

of $g(S)$ and $h(S)$. Thus, the first term can be estimated using the law of large numbers, and finite differences can be used to estimate the second term based on

$$\begin{aligned}
&-\partial_y E\left[g(S)\partial_\theta h(S)\cdot 1_{\{h(S)\geq y\}}\right]|_{y=0} \\
&\quad = -\lim_{\delta\to 0}\frac{1}{\delta}\{E\left[g(S)\partial_\theta h(S)\cdot 1_{\{h(S)\geq \delta/2\}}\right] - E\left[g(S)\partial_\theta h(S)\cdot 1_{\{h(S)\geq -\delta/2\}}\right]\} \\
&\quad = \lim_{\delta\to 0}\frac{1}{\delta}E[g(S)\partial_\theta h(S)\cdot 1_{\{-\delta/2\leq h(S)\leq \delta/2\}}] \\
&\quad = \lim_{\delta\to 0}\frac{1}{\delta}E\left[g(S)\partial_\theta h(S)\cdot Z\left(\frac{h(S)}{\delta}\right)\right],
\end{aligned}$$

where $Z(u) = 1_{\{-1/2\leq u\leq 1/2\}}$, which is known as the uniform or naive kernel because $u \sim U(-1/2, 1/2)$. It can be replaced with any other kernel, which we elaborate on in Section 5.3.3. In practice, Assumptions 1 and 2 are typically satisfied. Refer to [43] for the proof and a discussion of the assumptions.

5.3.2 Second-Order Greeks

The second-order Greeks are estimated using a theorem from [43] based on the following two assumptions:

Assumption 3. *For any $(\theta_1, \theta_2) \in \Theta$, $g(S)$ and $h(S)$ are differentiable with respect to θ_1 with probability 1 (w.p.1), and $\partial_{\theta_1} g(S)$ and $\partial_{\theta_1} h(S)$ are differentiable with respect to θ_2 with probability 1 (w.p.1), and there exist variables $K_g, K_h, L_g,$ and L_c with finite fourth moments, which may depend on (θ_1, θ_2), such that $|g(S(\theta_1 + \Delta\theta_1, \theta_2)) - g(S(\theta_1, \theta_2))| \leq K_g|\Delta\theta_1|$, $|g(S(\theta_1, \theta_2 + \Delta\theta_2,)) - g(S(\theta_1, \theta_2))| \leq K_g|\Delta\theta_2|$, $|h(S(\theta_1 + \Delta\theta_1, \theta_2)) - h(S(\theta_1, \theta_2))| \leq K_h|\Delta\theta_1|$, $|h(S(\theta_1, \theta_2 + \Delta\theta_2,)) - h(S(\theta_1, \theta_2))| \leq K_h|\Delta\theta_2|$, $|\partial_{\theta_1} g(S(\theta_1 + \Delta\theta_1, \theta_2)) - \partial_{\theta_1} g(S(\theta_1, \theta_2))| \leq L_g|\Delta\theta_2|$, and $|g(S(\theta_1, \theta_2 +$*

$\Delta\theta_2,))\partial_{\theta_1}h(S(\theta_1,\theta_2+\Delta\theta_2,)) - g(S(\theta_1,\theta_2))\partial_{\theta_1}h(S(\theta_1,\theta_2))| \leq L_c|\Delta\theta_2|$, *when* $|\Delta\theta_1|$ *and* $|\Delta\theta_2|$ *are sufficiently small.*

Assumption 4. *For any* $\theta \in \Theta$, $\partial_{\theta_1}\partial_{\theta_2}\phi(\theta_1,\theta_2,y)$ *exists and is continuous at* $(\theta_1,\theta_2,0)$, *where* $\phi(\theta_1,\theta_2,y) = E[g(S)\cdot 1_{\{h(S)\geq y\}}]$.

Theorem 5.3.2. *Suppose* $g(\cdot)$ *and* $h(\cdot)$ *satisfy Assumptions 3 and 4, and* $E[|g(S)|^4] < +\infty$ *and* $E[|h(S)|^4] < +\infty$. *Then*

$$\begin{aligned}\partial_{\theta_1}\partial_{\theta_2}p(\theta_1,\theta_2) &= E[\partial_{\theta_1}\partial_{\theta_2}g(S)\cdot 1_{\{h(S)\geq 0\}}] \\ &\quad -\partial_y E[(g(S)\partial_{\theta_1}\partial_{\theta_2}h(S) + \partial_{\theta_1}g(S)\partial_{\theta_2}h(S) + \partial_{\theta_2}g(S)\partial_{\theta_1}h(S))\cdot 1_{\{h(S)\geq y\}}]|_{y=0} \\ &\quad +\partial_y^2 E[g(S)\partial_{\theta_1}h(S)\partial_{\theta_2}h(S)1_{\{h(S)\geq y\}}]|_{y=0}.\end{aligned}$$

We focus on the second-order Greek gamma, where $\theta = S_0$; therefore,

$$\begin{aligned}\partial_\theta^2 p(\theta) &= E[\partial_\theta^2 g(S)\cdot 1_{\{h(S)\geq 0\}}] \\ &\quad -\partial_y E[(g(S)\partial_\theta^2 h(S) + 2\cdot\partial_\theta g(S)\partial_\theta h(S))\cdot 1_{\{h(S)\geq y\}}]|_{y=0} \\ &\quad +\partial_y^2 E[g(S)\partial_\theta h(S)\partial_\theta h(S)1_{\{h(S)\geq y\}}]|_{y=0}.\end{aligned}$$

Again, the first term on the right hand side can be estimated using using a sample mean. The finite differences derivation for the second term is similar to (5.3), and for the finite difference representation of the third term, observe that

$$\begin{aligned}&-\partial_y E[g(S)(\partial_\theta h(S))^2\cdot 1_{\{h(S)\geq y\}}]|_{y=0} \\ &\qquad = \lim_{\delta\to 0}\frac{1}{\delta}E\left[g(S)(\partial_\theta h(S))^2\cdot Z\left(\frac{h(S)-y}{\delta}\right)\right]\Big|_{y=0},\end{aligned}$$

and

$$\partial_y^2 E\left[g(S)(\partial_\theta h(S))^2 \cdot 1_{\{h(S)\geq y\}}\right]|_{y=0}$$
$$= \lim_{\delta\to 0}\frac{1}{\delta^2}E\left[g(S)(\partial_\theta h(S))^2 \cdot Z'\left(\frac{h(S)}{\delta}\right)\right], \quad (5.3)$$

where $Z(\cdot)$ is the kernel function. The approximation of the second-order auxiliary term (5.3) along with the first-order term (5.3) can be seen in the estimator (5.5).

5.3.3 Kernel

A d-dimensional kernel, $K : \mathbb{R}^d \to \mathbb{R}$, is a bounded symmetric density with respect to the the Lebesgue measure with $\lim_{\|u\|^d\to\infty} \| u \| K(u) = 0$ and $\int_{\mathbb{R}^d} \| u \|^2 K(u)du < \infty$ where $\| \cdot \|$ is any norm on $\mathbb{R}^d$ [6]. A kernel K has the form

$$K(u) = \frac{1}{h}K\left(\frac{u}{h}\right), u \in \mathbb{R}^d,$$

where h is the bandwidth, also known as the smoothing parameter [6]. For this paper, we focus on the case for $d = 1$. According to [6], a reasonable kernel does not affect the asymptotic behavior of the estimator; however, the bandwidth significantly influences the accuracy of the estimator. Large bandwidths decrease the variance but increase the bias, whereas small bandwidths lead to the exact opposite; hence, there is a tradeoff between variance and bias. Therefore, careful attention should be given to the selection process. Although there are other methods to quantify the accuracy of the kernel estimator, [43] measure it based on the MSE. Consequently, they attempt to select the bandwidth that maximizes the accuracy using an iterative pilot simulation.

5.3.4 First- and Second-Order Greek Estimators

Based on Theorem 5.3.1 and 5.3.2, the following estimators are used to approximate the first- and second-order Greeks, respectively:

$$\overline{G}_n = \frac{1}{n}\sum_{l=1}^{n} g_l^{'} \cdot 1_{[h_l \geq 0]} + \frac{1}{n\delta_n}\sum_{l=1}^{n} g_l \cdot h_l^{'} \cdot Z\left(\frac{h_l}{\delta_n}\right), \tag{5.4}$$

$$\overline{H}_n = \frac{1}{n}\sum_{l=1}^{n} g_l^{''} \cdot 1_{[h_l \leq 0]} + \frac{1}{n\delta_n}\sum_{l=1}^{n}\{g_l \cdot h_l^{''} + 2 \cdot g_l^{'} \cdot h_l^{'}\} \cdot Z\left(\frac{h_l}{\delta_n}\right) + \frac{1}{n\gamma_n^2}\sum_{l=1}^{n} g_l \cdot (h_l^{'})^2 \cdot Z^{'}\left(\frac{h_l}{\gamma_n}\right), \tag{5.5}$$

where $(g_l, h_l, g_l', h_l', g_l'', h_l'')$ is the lth observation of $(g(S), h(S), \partial_\theta g(S), \partial_\theta h(S), \partial_\theta^2 g(S), \partial_\theta^2 h(S)$, $Z(\cdot)$ is a kernel, and δ_n and γ_n are constant bandwidth parameters generated using a pilot simulation. The second-order estimator (5.5) is a special case when both derivatives are taken with respect to the same parameter.

5.4 Pilot Simulation

Ideally, the optimal bandwidth parameter would minimize the error between the Greek estimator and its true value, and the pilot simulation proposed in [43] employs an iterative method that generates an "optimal" bandwidth that minimize the asymptotic mean-squared error (MSE). The MSE of the first-order estimator can be expressed as

$$\begin{aligned} \mathrm{MSE}(\overline{G}_n) &= [E(\overline{G}_n) - \partial_\theta E[g(S) \cdot 1_{\{h(S)\geq 0\}}]]^2 + \mathrm{Var}(\overline{G}_n) \\ &= \delta_n^4 \left[\frac{\psi^{''}(0)}{2}\int_{-\infty}^{\infty} u^2 Z(u)du + \epsilon_n\right]^2 + \frac{1}{n\delta_n}(\sigma_1^2 + \xi_n), \end{aligned}$$

where $\psi(u) = f_h(u) \cdot E[g(S) \cdot \partial_\theta h(S) | h(S) = u]$, $\psi_\beta(u) = f_h(u) \cdot E[|g(S) \cdot \partial_\theta h(S)|^\beta | h(S) = u]$, and $\sigma_1^2 = \psi_2(0) \int_{-\infty}^{\infty} Z^2(u) du$. Since $\epsilon_n \to 0$ and $\xi_n \to 0$ as $n \to \infty$, then the optimal bandwidth choice is $c \cdot n^{-1/5}$, where

$$c = \left[\frac{\sigma_1^2}{(\psi''(0) \int_{-\infty}^{\infty} u^2 Z(u) du)^2} \right]^{1/5}.$$

This constant c is estimated by

$$\hat{c} = \left[\frac{\overline{V}_n}{(\widehat{\psi}''(0) \int_{-\infty}^{\infty} u^2 Z(u) du)^2} \right]^{1/5},$$

where $\overline{V}_n$ is the sample variance of $\overline{G}_n$ and $\widehat{\psi}''(0)$ is the finite difference approximation of $\psi''(0)$:

$$\widehat{\psi}''(0) = \frac{\overline{G}_n(s) + \overline{G}_n(-s) - 2\overline{G}_n(0)}{s^2},$$

where s is a sufficiently small step size and

$$\overline{G}_n(u) = \frac{1}{n\delta_n} \sum_{l=1}^{n} g_l \cdot h_l' \cdot Z\left(\frac{h(S) - u}{\delta_n} \right).$$

Therefore, the optimal bandwidth is approximated by $\delta_n^\star = \hat{c} \cdot n^{-1/5}$. The pilot simulation begins with $\hat{c} = 1$, and is updated after each iteration using the same sample. More specifically, at each iteration, $\overline{V}_n$ and $\widehat{\psi}''(0)$ are simulated using the updated $\delta_n^\star$ with $n = 500$. This process continues for a pre-specified number of iterations, and the bandwidth selected is the one generated after 30 pilot iterations, as in [43]. The pilot simulation algorithm does not specify the variance sample size n_v to generate $\overline{V}_n$, step size s to estimate $\psi''(0)$, and the number of iterations for the pilot simulation n_p, which are all predetermined. Refer to the e-companion of [43] for the algorithm details and the pilot simulation for the second-order Greeks.

5.5 Numerical Experiments

We conduct two sets of numerical experiments to test the robustness of a modified pathwise method and a pilot simulation used to generate an input parameter for the modified method [8]. The first is a sensitivity analysis of Greek estimators for both the Asian digital option and up-and-out barrier call option to bandwidth parameters, and the second explores the effect of the input parameters to the bandwidth generated from the pilot simulation. We considered identical settings as in [43], which are shown in Table 5.1 and described in Section 5.2. However, the sample size considered here is 100 as opposed to 1000 in [43]. From our preliminary results, the increase in sample size only tightens the confidence band and smooths out the curves, but the overall behavior is similar across estimators, so we opt to use a smaller sample size. Furthermore, we also consider the standard normal density as the kernel to increase the robustness of the resulting estimator.

5.5.1 Sensitivity to Bandwidth

In our experiment for each Greek, we generate 100 estimators for incrementally increasing bandwidths and observe the effect on the relative root mean-squared error (RRMSE) and the 95% confidence band (CB). Figures 5.1, 5.2, 5.3, and 5.4 plot the 95% confidence band for the Greek estimator generated using 1000 sample paths for Asian Greeks, 10000 sample paths for barrier Greeks, and sample size of 100 for incrementally increasing $\hat{c}$. The black curve in between is the mean, and the horizontal dashed line represents the exact Greek value. The three vertical lines in

Figures 5.1, 5.2, and 5.3 represent the 95% confidence interval for the $\hat{c}$ generated using the pilot simulation with $n_p = 30, n_v = 100$, which we will elaborate on in the next section. In the same figures, the second row of graphs plots the RRMSE of the 100 generated Greeks for incrementally increasing $\hat{c}$, where the additional vertical dashed line denotes the c^* that minimizes the RRMSE. The only difference when $n = 1000$ increases to $n = 10000$ for the Asian Greek estimators is the increased smoothness of the curves and tighter confidence bands, but again, the overall behavior is the same. [43] reports results for $n = 10^3, 10^4, 10^5$ for the Asian Greeks, but we only test the case for $n = 10^3$.

Not surprisingly, the bandwidth affects the accuracy and precision of the Greek estimator. Generally, as the bandwidth decreases, the variance increases, which can be seen in the widening of the 95% confidence band as $\hat{c} \to 0$ and is observable in all of the estimators. Similarly, the 95% confidence band decreases as the bandwidth increases, but eventually the confidence band no longer contains the exact Greek value. Figures 5.2, 5.3, and 5.4 for the Asian vega, Asian gamma, and barrier theta estimators illustrate this behavior, respectively. In addition, the minimum RRMSE of these estimates occur in one particular region for all discretization levels. Figure 5.2 shows that the Asian vega estimator is rather close to the exact value for a range of $\hat{c}$ values less than 0.1. The barrier theta estimator behaves similarly but for a different range of $\hat{c}$ values as seen in Figure 5.4. In contrast, the Asian gamma estimator is extremely sensitive to $\hat{c}$, and the RRMSE increases significantly with even slight deviations from the optimal bandwidth. Moreover, the 95% confidence interval (CI) for the bandwidth parameter is very narrow and does not contain the

Table 5.2: Asian delta.

n	k	c^*	RRMSE	lower	upper	range	δ_n^*	δ_n for s 0.1	0.01	0.001
500	10	0.64	0.028	0.915	1.021	0.106	0.185	0.14	0.06	0.06
	20	0.46	0.038	0.975	1.132	0.156	0.133	0.15	0.05	0.05
	50	0.54	0.033	0.998	1.130	0.132	0.156	0.14	0.06	0.06
1000	10	0.74	0.020	0.931	1.005	0.074	0.186	0.14	0.05	0.04
	20	0.54	0.028	0.996	1.104	0.108	0.136	0.13	0.05	0.05
	50	0.60	0.024	1.019	1.117	0.099	0.151	0.13	0.05	0.05
2000	10	0.84	0.014	0.943	0.996	0.053	0.184	0.13	0.03	0.03
	20	0.56	0.021	1.012	1.097	0.084	0.122	0.13	0.03	0.02
	50	0.68	0.017	1.034	1.105	0.071	0.149	0.13	0.03	0.02
5000	10	1.00	0.009	0.954	0.988	0.035	0.182	0.15	0.06	0.06
	20	0.66	0.014	1.027	1.083	0.055	0.120	0.16	0.07	0.07
	50	0.80	0.011	1.050	1.094	0.044	0.146	0.16	0.06	0.06
10000	10	1.14	0.007	0.960	0.985	0.020	0.181	0.15	0.03	0.02
	20	0.72	0.011	1.036	1.076	0.040	0.114	0.12	0.03	0.02
	50	0.92	0.008	1.056	1.088	0.032	0.146	0.13	0.03	0.02

exact Asian gamma value.

The numerical results for the Asian delta are quite different as illustrated in Figure 5.1. The mean of the delta estimator intersects with the exact/analytical value at least twice for $k = 10$ and $k = 50$. For the case $k = 10$, the RRMSE has a minimum and a nearly flat region. The minimum of the RRMSE for each discretization case occurs at different points for a given number of sample paths n, unlike the Asian vega, Asian gamma, and barrier theta. The exact $\hat{c}$ values at which the minimums occurs, the resulting RRMSE, and 95% confidence interval of the Asian delta estimators generated using the $\hat{c}$ are listed in Table 5.2.

Another interesting case is the gamma estimator for the up-and-out barrier call option, which relies on two bandwidth parameters δ_n and γ_n. The Asian gamma

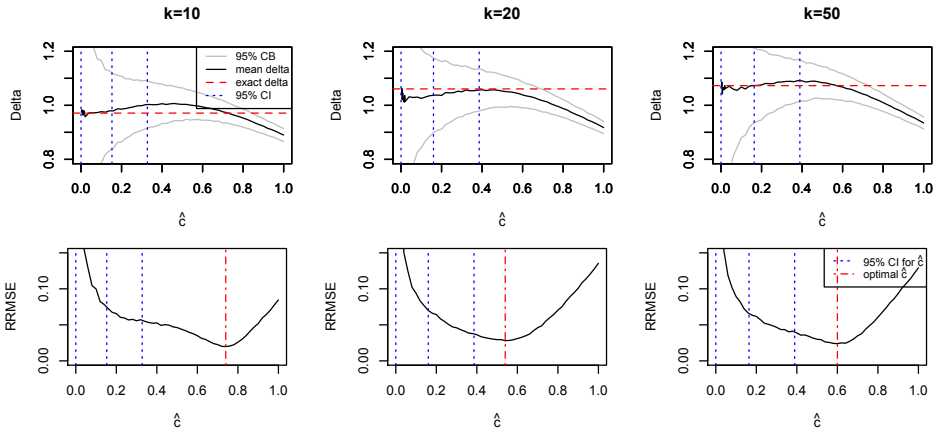


Figure 5.1: 95% confidence band for Asian delta (solid curves) for $n = 1000$ with w/95% confidence interval for bandwidth (vertical dashed lines) generated using $s = 0.1$ and RRMSE for 100 sample paths.

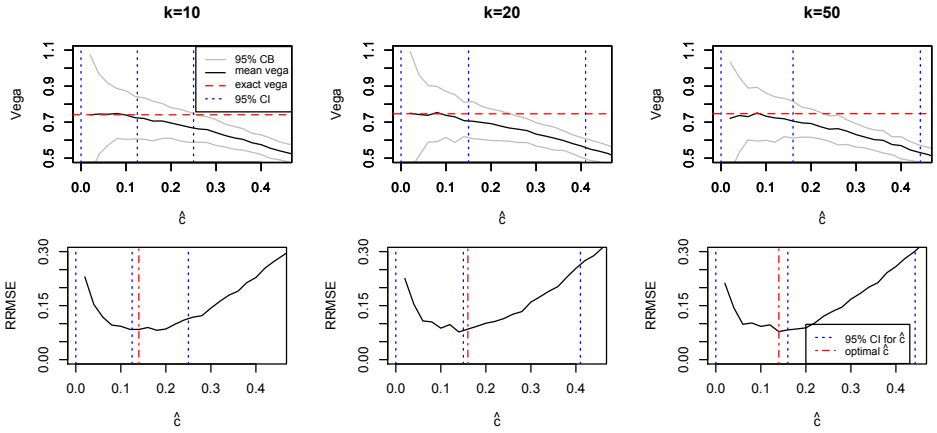


Figure 5.2: 95% confidence band for Asian vega (solid curves) for $n = 1000$ w/95% confidence interval for bandwidth (vertical dashed lines) generated using $s = 0.001$ and RRMSE for 100 sample paths.

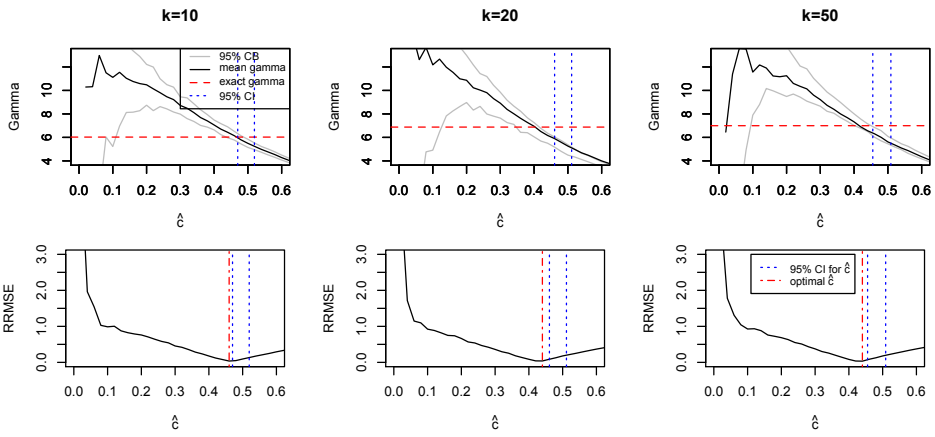


Figure 5.3: 95% confidence band for Asian gamma (solid curves) for $n = 1000$ w/95% confidence interval for bandwidth (vertical dashed lines) generated using $s = 0.01$ and RRMSE for 100 sample paths.

is also a second-order Greek, but the second term in (5.4) containing δ_n disappears with the given settings. Figure 5.5 plots the barrier gamma as a function of the bandwidth parameters δ_n (visible horizontal axis) and γ_n (not visible), where the vertical planes represent the approximately exact gamma for $k = 20, 50$, respectively. It is clear that gamma is insensitive to γ_n, but extremely sensitive to δ_n, especially when $\delta_n < .03$. Unfortunately, this is the region in which the estimator attains the value closest to the true value. Since the analytical expression is not available for comparison, the true values are approximated using finite differences with a large sample of 10^9. In this case, the steepness of the gamma estimator near the "exact" value prompts the need for a very accurate and precise δ_n; otherwise, the estimate could be drastically different from its true value. As expected, the margin of error decreases as the number of sample paths n increases. For each discretization level k, $\hat{c}$ increases with n, but the resulting δ_n values are similar for each k regardless of

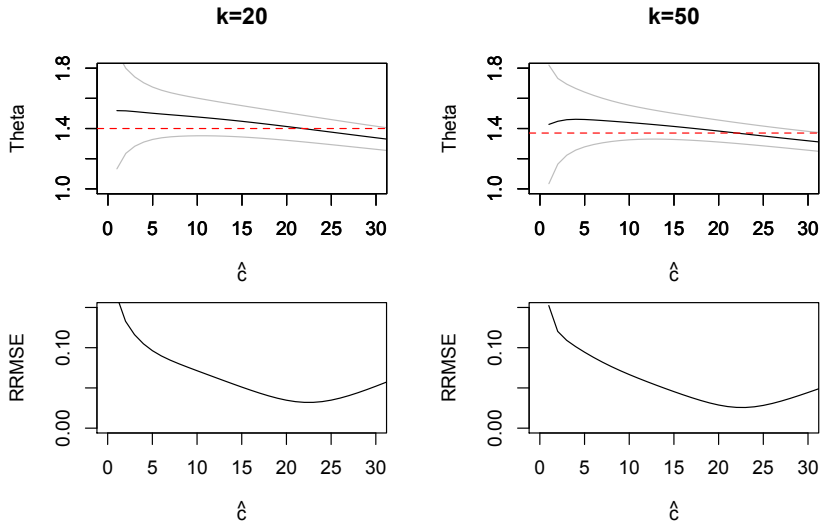


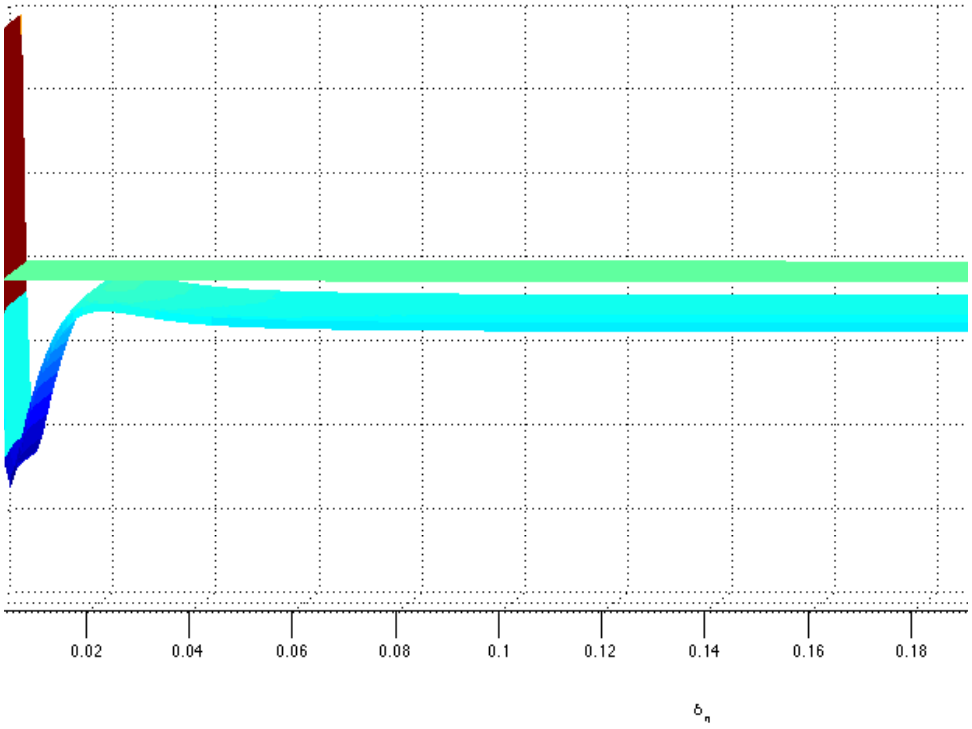
Figure 5.4: 95% confidence band for barrier theta for $n = 10000$ and RRMSE for 100 sample paths.

n.

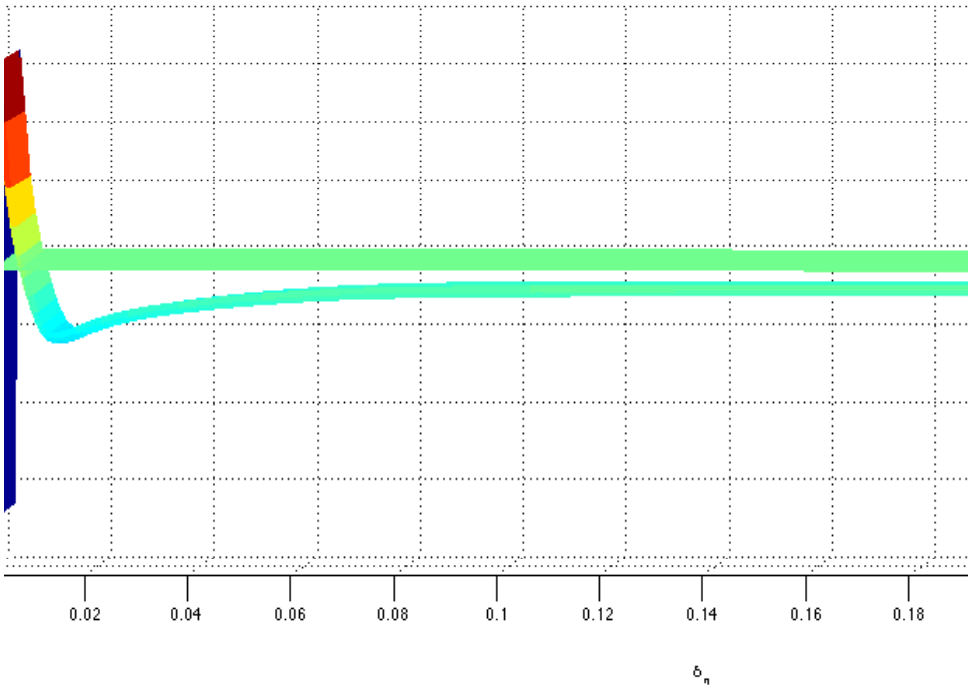
The conclusions in this section are representative of all of the estimators tested. For instance, the Asian delta is the only estimator whose mean matches the exact Greek value for two different bandwidth values and the barrier gamma is the only estimator that requires the input of two bandwidth parameters. The Asian vega, in general, has a decreasing 95% confidence band whose mean also decreases with $\hat{c}$ and matches the exact Greek value for one particular $\hat{c}$, and there is one global minimum RRMSE, which represents the results of the remaining Greek estimators.

5.5.2 Sensitivity of Bandwidth Generator to Input Parameters

Our sensitivity analysis of the modified pathwise method applied to Greeks concludes that it is, in general, very sensitive to the bandwidth parameter(s), so the key to a good estimator is in the bandwidth selection process. Therefore, we inves-



(a) $k = 20$



(b) $k = 50$

Figure 5.5: Barrier gamma estimator for $n = 10000$, 100 sample paths.

tigate the sensitivity of the pilot simulation to the various input parameters such as variance sample size n_v, perturbation size s, number of sample paths n, and number of pilot iterations n_p, as described in Section 5.4. We considered the following settings: $n_v \in \{50, 100, 500, 1000\}$, $s \in \{0.001, 0.01, 0.1\}$, $n \in \{500, 1000, 2000, 5000\}$, and $n_p \in \{1, \ldots, 100\}$. For each setting, we compute the 95% confidence interval for the bandwidth output and compare it against the optimal bandwidth that minimizes the RRMSE in Table 5.2. The pilot simulation experiments consider a variance sample size of 100 (i.e., $n_v = 100$) and 30 pilot iterations (i.e, $n_p = 30$), unless stated otherwise. Over an extensive set of parameter settings, the results are quite similar, so we limit our discussion to representative cases.

First, we discuss the results of the pilot simulation for the Asian delta. For any fixed perturbation size, the average $\hat{c}$ under the Asian delta estimator is quite similar across sample paths and discretization level k, with a maximum range of 0.04. As n increases, the average $\hat{c}$ decreases ever so slightly. One would think that variance of $\hat{c}$ decreases with n, i.e., the confidence band would decrease with the n; however, the perturbation size s appears to be the determining factor from Figure 5.6. In this case, the smaller perturbation size $s = 0.1$ leads to wider confidence intervals, which is interesting, because smaller perturbations tend to generate noisier estimates. Therefore, it could be advantageous to consider an average or a window average, as opposed to selecting the $\hat{c}$ generated after a certain number of iterations. For instance, the average $\hat{c}$ fluctuates, especially in the case of $s = 0.1$, so taking a further average of multiple pilot iterations at the later stage, could generate a parameter that is more stable.

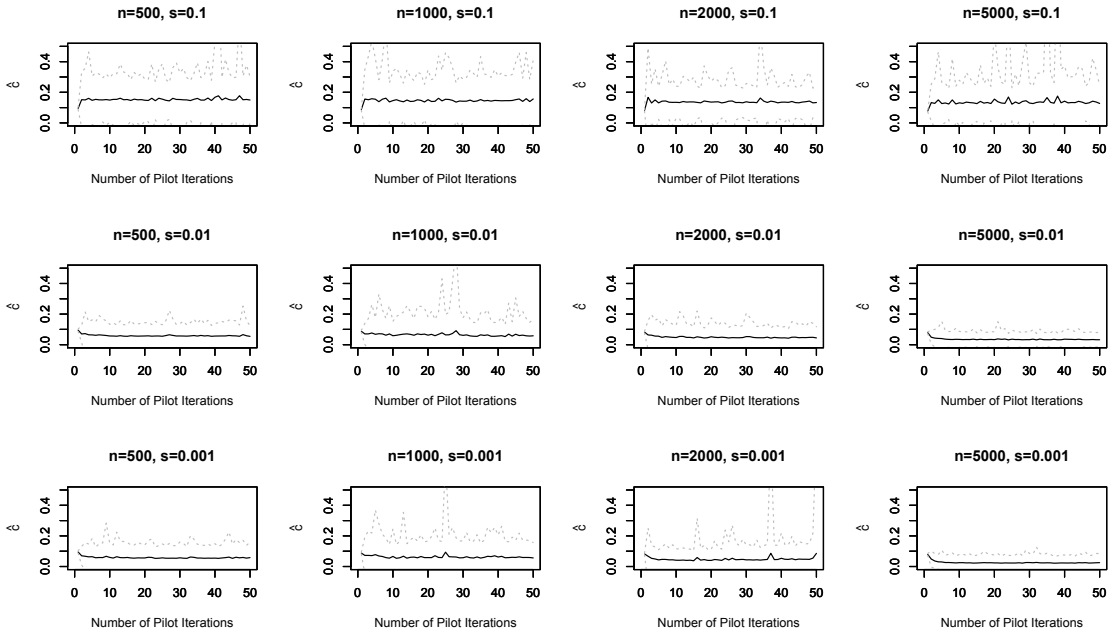


Figure 5.6: Asian delta pilot 95% confidence interval for $\hat{c}$ for $k = 10$.

Another input parameter that was not specified is the sample size n_v used to generate the sample variance $\overline{V}_n$. So we investigate the sensitivity of the bandwidth parameter to n_v for each of the perturbation sizes s. Again, we focus on the Asian delta case for the discretization case $k = 10$, sample path $n = 1000$, vary $n_v \in \{50, 100, 500, 1000\}$, and $s \in \{0.001, 0.01, 0.1\}$. Figure 5.7 plots the generated $\hat{c}$ against the number of pilot simulations n_p for a sample size of 100. The results of the sensitivity to n_v are inconclusive, since the behavior is inconsistent across different parameter settings. For this particular example, the larger perturbation size $s = 0.1$ leads to a higher bandwidth parameter, which can also be seen in Figure 5.6.

Clearly, the input parameters have a significant impact on the bandwidth

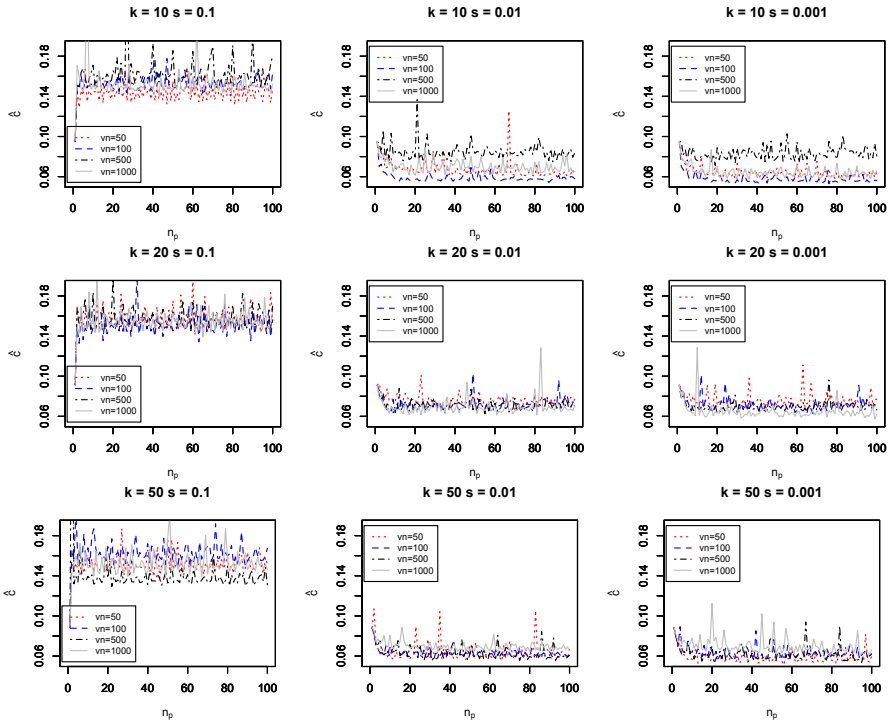


Figure 5.7: Asian delta pilot, $n = 500$, sample size $= 100$.

parameter generated from the pilot simulation, and we investigate further to determine whether or not the confidence interval of the bandwidth parameter captures the bandwidth that minimizes the RRMSE for parameters $n = 1000$, $n_v = 100$, $n_p = 30$, sample size 100, $s = 0.1$ for the delta, $s = 0.01$ for the gamma, and $s = 0.001$ for both vega and theta Asian Greek estimators. Values for the perturbation parameters were not specified in [43], but were obtained from the authors via personal communication. The sizes were chosen relative to the parameter of interest. Our results vary significantly, from successfully capturing the optimal bandwidth to completely missing it. Figures 5.1, 5.2, and 5.3 contain vertical dotted lines representing the 95% confidence interval for the optimal c^* with the above specified parameters, the dotted line denotes the mean, and the dashed line is the c^* that

minimizes the RRMSE from our first set of numerical experiments. None of the 95% confidence intervals of $\hat{c}$ capture the minimum RRMSE for the Asian delta Greek estimator. Instead, the CI contains mean $\hat{c}^*$. The confidence interval for the $\hat{c}$ generated from the pilot simulation for the Asian vega, covers the minimum RRMSE bandwidth, which can be seen in Figure 5.2.

5.6 Summary and Future Work

In this chapter, we analyzed the sensitivity of the Greek kernel estimators of [43] to the bandwidth parameters and the sensitivity and performance of the pilot simulation in selecting the "optimal" bandwidth parameters to input parameters. The simulated experiments show that the kernel estimators are quite sensitive to the bandwidth choice, and the pilot simulation is sensitive to the input parameters. Of the input parameters, the perturbation step size s significantly impacts the generated bandwidth. If s is too small relative to the parameter of interest, then the pilot simulation could generate bandwidths that lead to poor gradient estimates. However, even if the s is chosen proportional to the parameter of interest, the 95% confidence interval for the $\hat{c}$ might not capture the c^* that minimizes the RRMSE. The input parameters play a vital role in the performance of the estimator, so the input selection process of the pilot simulation and other bandwidth selection methods are important future research areas.

Bibliography

[1] S. Andradóttir. A new algorithm for stochastic approximation. *Proceedings of the 1990 Winter Simulation Conference*, pages 364–366, 1990.

[2] S. Andradóttir. A stochastic approximation algorithm with varying bounds. *Operations Research*, 43(6):1037–1048, 1995.

[3] S. Andradóttir. A scaled stochastic approximation algorithm. *Management Science*, 42(4):475–498, 1996.

[4] S. Asmussen and P. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Springer, New York, 2007.

[5] J. Blum. Multidimensional stochastic approximation methods. *The Annals of Mathematical Statistics*, 25(4):737–744–200, 1954.

[6] D. Bosq. *Nonparametric Statistics for Stochastic Processes*. Springer, second edition, 1998.

[7] 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.

[8] M. Chau and M. Fu. On the sensitivity of Greek kernel estimators to bandwidth parameters. *Proceedings of the 2014 Winter Simulation Conference*, pages 3845–3856, 2014.

[9] M. Chau and M. Fu. An overview of stochastic approximation. In M. Fu, editor, *Handbook of Simulation Optimization*, chapter 6, pages 149–178. Springer, 2014.

[10] M. Chau, H. Qu, and M. Fu. A new hybrid stochastic approximation algorithm. In *Proceedings of the 2014 Workshop on Discrete Event Systems*, volume 12, pages 241–246, 2014.

[11] M. Chau, H. Qu, and M. F. I. Ryzhov. An empirical sensitivity analysis of the Kiefer-Wolfowitz algorithm and its variants. *Proceedings of the 2013 Winter Simulation Conference*, pages 945–965, 2013.

[12] H. Chen, T. Duncan, and B. Pasik-Duncan. A Kiefer-Wolfowitz algorithm with randomized differences. *IEEE Transactions on Automatic Control*, 44(3):442–453, 1999.

[13] H. Chen and Y. Zhu. Stochastic approximation procedure with randomly varying truncations. *Scientia Sinica Series A*, 29:914–926, 1986.

[14] K. Chung. On a stochastic approximation method. *The Annals of Mathematical Statistics*, 25(3):463–483, 1954.

[15] B. Delyon and A. Juditsky. Accelerated stochastic approximation. *SIAM Journal on Optimization*, 3(4):868–881, 1993.

[16] 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.

[17] J. Dippon and J. Renz. Weighted means in stochastic approximation of minima. *SIAM Journal on Control Optimization*, 35(5):1811–1827, 1997.

[18] V. Dupac. On the Kiefer-Wolfowitz approximation method. *Časopis pro péstování Matematiky*, 082(1):47–75, 1957.

[19] V. Fabian. Stochastic approximation of minima with improved asymptotic speed. *The Annals of Mathematical Statistics*, 38(1):191–200, 1967.

[20] V. Fabian. On asymptotic normally in stochastic approximation. *The Annals of Mathematical Statistics*, 39(4):1107–1380, 1968.

[21] V. Fabian. Stochastic approximation. In J. Rustagi, editor, *Optimizing Methods in Statistics*, pages 439–470. Academic Press, New York, 1971.

[22] E. Fogel. A fundamental approach to the convergence analysis of least squares algorithms. *IEEE Transactions on Automatic Control*, 26(3):646 – 655, 1981.

[23] M. Fu. Gradient estimation. In S. Henderson and B. Nelson, editors, *Handbooks in Operations Research and Management Science: Simulation*, chapter 19, pages 575–616. Elsevier, Amsterdam, 2006.

[24] M. Fu. What you should know about simulation and derivatives. *Naval Research Logistics*, 55(8):723 – 736, 2008.

[25] M. Fu. Stochastic gradient estimation. In M. Fu, editor, *Handbook of Simulation Optimization*, chapter 5, pages 105–147. Springer, 2014.

[26] M. Fu and S. Hill. Optimization of discrete event systems via simultaneous perturbation stochastic approximation. *IIE Transactions*, 29(3):233–243, 1997.

[27] M. Fu and Y. Ho. Using perturbation analysis for gradient estimation, averaging and updating in a stochastic approximation algorithm. *Proceedings of the 1988 Winter Simulation Conference*, pages 509–517, 1988.

[28] M. Fu and J. Hu. *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*. Kluwer, Boston, MA, 1997.

[29] A. George and W. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65(1):167–198, Oct. 2006.

[30] 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.

[31] 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.

[32] 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.

[33] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, New York, NY, 2004.

[34] Y. Ho and X. Cao. Perturbation analysis and optimization of queueing networks. *Journal of Optimization Theory and Applications*, 40(4):550–582, 1983.

[35] Y. Ho, L. Shi, L. Dai, and W.-B. Gong. Optimizing discrete event dynamic systems via the gradient surface method. *Discrete Event Dynamic Systems: Theory and Applications*, 2(2):99–120, 1992.

[36] H. Kesten. Accelerated stochastic approximation. *The Annals of Mathematical Statistics*, 29(1):41–59, 1958.

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

[38] N. Kleinman, J. Spall, and D. Naiman. Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, 45(11):1570–1578, 1999.

[39] H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer, New York, NY, 1987.

[40] H. 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.

[41] H. Kushner and J. Yang. Stochastic approximation with averaging and feedback: Rapidly convergent 'on-line' algorithms. *IEEE Transactions on Automatic Control*, 40:24–34, 1995.

[42] H. Kushner and G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, New York, NY, 2003.

[43] G. Liu and L. Hong. Kernel estimation of the Greeks for options with discontinuous payoffs. *Operations Research*, 59(1):96–108, 2011.

[44] J. Maryak. Some guidelines for using iterate averaging in stochastic approximation. *Proceedings of the 36th IEEE Conference on Decision and Control*, 3:2287–2290, 1997.

[45] 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.

[46] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley, New York, NY, 1983.

[47] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[48] B. Polyak. New stochastic approximation type procedures. *Automat. i Telemekh*, 51:98–105, 1990.

[49] B. Polyak and A. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[50] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

[51] D. Ruppert. A Newton-Raphson version of the multivariate Robbins-Monro procedure. *The Annals of Statistics*, 13(1):236–245, 1985.

[52] D. Ruppert. Efficient estimations from a slowly convergent Robbins-Monro process. Technical report, Cornell University Operations Research and Industrial Engineering, Ithaca, NY, February 1988.

[53] J. Sacks. Asymptotic distribution of stochastic approximation procedures. *The Annals of Mathematical Statistics*, 29(2):351–634, 1958.

[54] P. Sadegh. Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation. *Automatica*, 33(5):889–892, 1997.

[55] J. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[56] J. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10):1839–1853, 2000.

[57] J. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley, Hoboken, NJ, 2003.

[58] A. Tsybakov and B. Polyak. Optimal order of accuracy of search algorithms in stochastic optimization. *Problemy Peredachi Informatsii*, 26(2):45–63, 1990.

[59] J. Venter. An extension of the Robbins-Monro procedure. *The Annals of Mathematical Statistics*, 38(1):181–190, 1967.

[60] I. Wang and E. Chong. A deterministic analysis of stochastic approximation with randomized differences. *IEEE Transactions on Automatic Control*, 43(12):1745–1749, 1998.

[61] Y. Wang, M. Fu, and S. Marcus. Sensitivity analysis for barrier options. In M. Rossetti, R. Hill, B. Johansson, and R. Ingalls, editors, *Proceedings of the 2009 Winter Simulation Conference*, pages 1272 – 1282, Piscataway, New Jersey, 2009. Institute of Electrical and Electronics Engineers, Inc.

[62] M. Wasan. *Stochastic Approximation*. Cambridge Tracts Mathematics and Mathematical Physics. Cambridge University Press, London, 1969.

[63] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning*, 11:2543–2596, 2010.

[64] G. Yin and Y. Zhu. Almost sure convergence of stochastic approximation algorithms with nonadditive noise. *International Journal of Control*, 49(4):1361–1376, 1989.

[65] F. Yousefian, A. Nedíc, and U. Shanbhag. On stochastic gradient and subgradient methods with adaptive steplength sequences. *Automatica*, 48(1):56–67, 2011.