CYCLIC STOCHASTIC OPTIMIZATION: GENERALIZATIONS, CONVERGENCE, AND APPLICATIONS IN MULTI-AGENT SYSTEMS

by
Karla Hernández Cuevas

A dissertation submitted to The Johns Hopkins University
in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland
August, 2017

Abstract

Stochastic approximation (SA) is a powerful class of iterative algorithms for nonlinear root-finding that can be used for minimizing a loss function, $L(\theta)$, with respect to a parameter vector θ, when only noisy observations of $L(\theta)$ or its gradient are available (through the natural connection between root-finding and minimization); SA algorithms can be thought of as stochastic line search methods where the entire parameter vector is updated at each iteration. The cyclic approach to SA is a variant of SA procedures where θ is divided into multiple subvectors that are updated one at a time in a cyclic manner.

This dissertation focuses on studying the asymptotic properties of cyclic SA and of the generalized cyclic SA (GCSA) algorithm, a variant of cyclic SA where the subvector to update may be selected according to a random variable or according to a predetermined pattern, and where the noisy update direction can be based on the updates of any SA algorithm (e.g., stochastic gradient, Kiefer–Wolfowitz, or simultaneous perturbation SA). The convergence of GCSA, asymptotic normality of GCSA (related to rate of convergence), and ef-

ficiency of GCSA relative to its non-cyclic counterpart are investigated both analytically and numerically. Specifically, conditions are obtained for the convergence with probability one of the GCSA iterates and for the asymptotic normality of the normalized iterates of a special case of GCSA. Further, an analytic expression is given for the asymptotic relative efficiency (when efficiency is defined in terms of mean squared error) between a special case of GCSA and its non-cyclic counterpart. Finally, an application of the cyclic SA scheme to a multi-agent stochastic optimization problem is investigated.

This dissertation also contains two appendices. The first appendix generalizes Theorem 2.2 in Fabian (1968) (a seminal paper in the SA literature that derives general conditions for the asymptotic normality of SA procedures) to make the result more applicable to some modern applications of SA including (but not limited to) the GCSA algorithm, certain root-finding SA algorithms, and certain second-order SA algorithms. The second appendix considers the problem of determining the presence and location of a static object within an area of interest by combining information from multiple sensors using a maximum-likelihood-based approach.

Primary Reader and Advisor: James C. Spall

Second Reader: Raman Arora

To my parents.

Acknowledgments

I would first like to thank my advisor, James Spall, for his invaluable guidance throughout my graduate studies. His dedication to his students, patience, admirable work ethic, and vast knowledge were instrumental in making my experience as a graduate student fruitful, enjoyable, and enriching.

I would also like to give special thanks to Stephen Lee, my husband and best friend, for his friendship, support, and for the *many* interesting discussions we've shared (of a mathematical and non-mathematical nature) throughout our time together. I would also like to mention that the proof of Lemma 11 was the result of a discussion with Stephen.

I would also like to thank fellow graduate student Jingyi Zhu (currently a Ph.D. student at the department of Applied Mathematics & Statistics at Johns Hopkins) for reading Chapter 4 of my dissertation and for her detailed feedback which helped me improve the presentation of the chapter.

I would also like to express my gratitude to Raman Arora, Danial Naiman, Daniel Robinson, Maxim Bichuch, and John Wierman, the members of my

dissertation defense committee, for their time with special thanks to Raman Arora, my second reader; reviewing such a lengthy dissertation is no small task and I sincerely appreciate his help.

I also want to thank all the professors whose courses I've had the pleasure of taking. Although I have taken several courses during my studies at Hopkins, the courses taught by professors James Fill, James Spall, and John Wierman stand out as having been instrumental to my research.

Last but not least, I would like to thank my family with special thanks to my parents. Without their support I would not be where I am today.

This work was financially supported by the National Council of Science and Technology (CONACYT) of Mexico, the Office of Naval Research via Navy contract N00024-13-D6400, the Acheson J. Duncan Fund for the Advancement of Research in Statistics, and Dr. James Spall's JHU/APL sabbatical professorship at Johns Hopkins' Whiting School of Engineering.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The objective of unconstrained optimization problems is to minimize a real-valued loss function $L(\theta)$ with respect to a parameter vector $\theta \in \mathbb{R}^p$. An important limitation in many practical problems is the fact that the loss function itself is unknown in the sense that $L(\theta)$ may only be observable in the presence of noise. For example, suppose that the loss function represents the expected output of a complex stochastic system that depends on θ. In this case, obtaining a closed form expression for $L(\theta)$ (or even evaluating $L(\theta)$ at a given θ) may be impossible since computing the expected output would require detailed knowledge of the stochastic process governing the output of the system. In this light, this dissertation makes a distinction between deterministic optimization,

where the loss function is known and the optimization process is entirely deterministic, and stochastic optimization, where the optimization process involves some type of randomness (e.g., the optimization algorithm uses measurements of $L(\theta)$ that are corrupted by random noise or there is a random choice made in the search direction as the algorithm iterates towards a solution). Stochastic approximation (SA) is a powerful class of iterative algorithms for nonlinear root-finding. These algorithms can also be used for stochastic optimization (through the natural connection between root-finding and minimization). The cyclic approach to SA is a particular variant of these iterative procedures in which only a subset of the parameter vector (referred to as a *subvector*) is updated at any given time. This dissertation focuses on studying the asymptotic properties (e.g., convergence and rate of convergence) of cyclic SA (and generalizations) for stochastic optimization.

In the cyclic approach, the full parameter vector is divided into two or more subvectors and the process proceeds by sequentially updating each of the subvectors, while holding the remaining parameters at their most recent values. Thus, the cyclic approach is iterative in nature and at each iteration an estimate for the minimizer of $L(\theta)$ is obtained. To illustrate the idea behind the cyclic approach, let $\hat{\theta}_k$ denote the estimate obtained during the kth iteration. Furthermore, consider the special case of the cyclic approach where there are only two subvectors, $\hat{\theta}_k^{[1]}$ and $\hat{\theta}_k^{[2]}$, so that $\hat{\theta}_k = [(\hat{\theta}_k^{[1]})^\top, (\hat{\theta}_k^{[2]})^\top]^\top$. Additionally,

for simplicity assume the two subvectors are updated in a strictly-alternating manner. Here, the first step of an iteration of the cyclic approach would consist of updating $\hat{\theta}_k^{[1]}$ while keeping $\hat{\theta}_k^{[2]}$ fixed. This would give rise to the vector $[(\hat{\theta}_{k+1}^{[1]})^\top, (\hat{\theta}_k^{[2]})^\top]^\top$. In the second step $\hat{\theta}_k^{[2]}$ would be updated while holding $\hat{\theta}_{k+1}^{[1]}$ fixed, this would give rise to the vector $\hat{\theta}_{k+1} = [(\hat{\theta}_{k+1}^{[1]})^\top, (\hat{\theta}_{k+1}^{[2]})^\top]^\top$.

Independent of whether an optimization algorithm is stochastic or deterministic, cyclic schemes can help reduce the problem's complexity by focusing only on a subset of the parameter vector at any given time. For example, conditional optimization with respect to subsets of θ may prove more tractable than simultaneous (unconditional) optimization. Lee and Park (2008), for example, use cyclic optimization to approach a structure-from-motion problem (presented as a deterministic optimization problem), where minimizing $L(\theta)$ is difficult but where it is possible to obtain closed form expressions for the minimizers $L(\theta)$ with respect to different subsets of θ. Lee and Park (2008) propose an algorithm that consists of sequentially minimizing the loss function with respect to the different subsets of θ. Although the authors do not provide any theoretical guarantees, the algorithm appears to exhibit good numerical performance. Another example lies in the area of communication networks. Here, one may be interested in jointly optimizing congestion control and scheduling (see, for example, Andrews 2006) and it may be the case that conditional solutions are more readily available.

In the cyclic approach, the full parameter vector is divided into two or more subvectors and the process proceeds by sequentially optimizing each of the subvectors, while holding the remaining parameters at their most recent values. The cyclic approach is of special interest where there is need to extend a model to include new parameters since it allows for the preservation of resources dedicated to optimizing the original problem (e.g., the preservation of expensive software). In theory, one could alternate between updating the new parameters and the parameters from the original problem (existing resources could be reused with this approach).

A convenient feature of cyclic line search methods like block coordinate descent is that computing partial update directions (e.g., computing only a part of the gradient when the update directions are gradient-based) is often significantly less expensive than computing the full update direction. In this vein, Wright (2015) discusses how coordinate descent algorithms have grown in popularity because of their usefulness in data analysis, and machine learning. Wright (2015) gives a useful literature review of coordinate descent methods that are mostly limited to deterministic optimization algorithms, with the exception of randomized coordinate descent methods which can be seen as a special case of the stochastic gradient algorithm (a type of SA algorithm).

In the area of multi-agent optimization, the vector θ can be divided into different subsets each of which is updated by a different agent (see, for exam-

ple, Peterson et al. 2014 and Botts et al. 2016, where θ is a vector related to the positions of the agents). In general, agents may operate in a synchronous manner (i.e., agents synchronize the timing of their respective updates in some way, such as by taking turns performing updates) or in an asynchronous manner (i.e., agents do not synchronize the timing of their respective updates in any way). Cyclic optimization algorithms apply given the manner in which agents operate and, under appropriate conditions, allow for rigorous theoretical analysis of multi-agent optimization algorithms.

Despite the aforementioned desirable properties of cyclic optimization, convergence to a minimizer of $L(\theta)$ (either global or local) is not generally guaranteed for cyclic algorithms. To illustrate this fact, consider an example where $L(\theta) = -(\tau_1+\tau_2)$ and $\theta \in \mathcal{S} \equiv \{[\tau_1, \tau_2]^\top$such that $\tau_1 \in [0, 1]$ and $\tau_2 \in [0, 2(1 - \tau_1)]\}$ (this set defines a region bounded by a right triangle with base corresponding to the interval $[0, 1]$ on the τ_1-axis, right angle at $[0, 0]^\top$, and height 2). This constrained optimization problem has a unique global minimum at $\theta^* = [0, 2]^\top$. Suppose now that a cyclic algorithm is implemented given that one alternates between minimizing $L(\theta)$ with respect to τ_1 and τ_2, beginning with τ_1. Furthermore, suppose the algorithm is initialized at $\hat{\theta}_0 = [0, 0]^\top$. In its first step, the algorithm will arrive at the point $\hat{\theta}_1 = [1, 0]^\top$. Afterwards, the algorithm becomes "stuck" at this point since no update in the value of τ_2 can further educe the loss function value. Thus, this cyclic algorithm would never reach the global

minimum at θ^*. Moreover, the point $\hat{\theta}_1 = [1, 0]^\top$ is not even a *local* minimum. This can be seen by noting that the point $\theta_\epsilon = [1 - \epsilon, 2\epsilon]^\top$ with $\epsilon \in [0, 1]$ is in the set $\mathcal{S}$ and that $L(\theta_\epsilon) = -(1 + \epsilon) < -1 = L(\hat{\theta}_1)$. For both deterministic- and stochastic optimization algorithms, knowing that a non-cyclic algorithm converges to a minimizer (either local or global) of $L(\theta)$ does not automatically imply that their cyclic counterparts also converge to a minimizer or $L(\theta)$.

This dissertation introduces the generalized cyclic SA (GCSA) algorithm, a cyclic algorithm where the subvectors are updated using SA. Loosely speaking, at each time increment a subvector of the parameter vector is selected and updated according to a direction that is obtained using SA (this includes many popular stochastic update directions including the those used in the Robbins–Monro, Kiefer–Wolfowitz, and simultaneous perturbation SA algorithms). In the GCSA algorithm the subvector to update is not necessarily selected following a deterministic pattern (the term "generalized" in "generalized cyclic SA" is used precisely to emphasize this fact). A special case of the GCSA algorithm, for example, allows the subvector that is to be updated to be selected according to a random variable that may depend on the iteration number. The following section reviews the literature for existing results on cyclic optimization. We cover both deterministic and stochastic implementations.

Before surveying the literature we first review the basic form of SA algorithms for nonlinear root-finding. This allows us to be more specific when com-

paring existing results to the results developed in this dissertation. Given a vector θ and a vector-valued function $\boldsymbol{f}(\theta)$, the basic SA algorithm for nonlinear root-finding attempts to find a solution to $\boldsymbol{f}(\theta) = 0$ in an iterative manner using the recursion $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{Y}_k(\hat{\theta}_k)$, where $a_k > 0$ is the gain sequence and $-\boldsymbol{Y}_k(\hat{\theta}_k)$ is a vector-valued random variable representing a noisy observation of $-\boldsymbol{f}(\hat{\theta}_k)$, the desired update direction. In the analysis of SA algorithms it is often useful to express the vector $\boldsymbol{Y}_k(\hat{\theta}_k)$ as follows:

$$\boldsymbol{Y}_k(\hat{\theta}_k) = \boldsymbol{f}(\hat{\theta}_k) + \beta_k(\hat{\theta}_k) + \xi_k(\hat{\theta}_k). \tag{1.1}$$

Typically, $\beta_k(\hat{\theta}_k) \equiv E\big[\boldsymbol{Y}_k(\hat{\theta}_k) - \boldsymbol{f}(\hat{\theta}_k)\big|\mathcal{F}_k\big]$, $\xi_k(\hat{\theta}_k) \equiv \boldsymbol{Y}_k(\hat{\theta}_k) - E\big[\boldsymbol{Y}_k(\hat{\theta}_k)\big|\mathcal{F}_k\big]$, $\mathcal{F}_k$ is some representation of the history of the process, and $E[\mathcal{X}]$ represents the expected value of the random variable $\mathcal{X}$. In the special case where $\boldsymbol{f}(\theta) = \boldsymbol{g}(\theta)$ is the gradient of $L(\theta)$, that is when SA is used for stochastic optimization via nonlinear root-finding, $\boldsymbol{Y}_k(\hat{\theta}_k)$ denotes a noisy estimate of the gradient. Here, it is common to replace the notation $\boldsymbol{Y}_k(\hat{\theta}_k)$ with $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$. Therefore, we write $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\theta}_k)$, (Section 2.2 discusses SA in greater detail).

1.2 Literature Review

In the 1980s, 1990s, and early 2000s, Bertsekas and Tsitsiklis (1989), Luo and Tseng (1992), Luo and Tseng (1993), and Tseng (2001) made important

contributions to understanding the convergence properties of cyclic optimization procedures. However, their analysis focused on deterministic optimization problems, not on the stochastic optimization setting considered in this dissertation. A few more recent references investigating cyclic implementations in the area of deterministic optimization are Tseng and Yung (2009), who investigate the convergence of block-coordinate gradient descent methods for linearly constrained non-smooth separable loss functions, and Spall (2012), who investigates the convergence properties of general cyclic algorithms. The interested reader can also find a useful literature review of coordinate descent methods (largely for deterministic optimization) in Wright (2015). A few applications of cyclic procedures for deterministic optimization can be found in the papers by Canutescu and Dunbrack (2003), who use cyclic procedures for a control problem in robotics, Lee and Park (2008), who use cyclic optimization for a problem in computer vision, Li and Osher (2009), who consider a compressed sensing problem, and Li and Petropulu (2014), who use the alternating directions method of multipliers (a cyclic procedure) for target location estimation.

The previous references are concerned with cyclic schemes for solving deterministic optimization problems. In the area of stochastic optimization, one class of SA algorithms which at first appears to be related to GCSA is the class of two time-scales SA algorithm studied in Borkar (1997) (see also Konda and Tsitsiklis 2004 for a similar setting). Here, a parameter vector, $\boldsymbol{X}(k)$, is

updated using SA. In Borkar's (1997) formulation, obtaining the SA update direction requires a second SA step. Specifically, using Borkar's (1997) notation, there exists a vector $\boldsymbol{U}(k)$ such that:

$$\boldsymbol{X}(k+1) = \boldsymbol{X}(k) + a(k)\left[\boldsymbol{R}(\boldsymbol{X}(k), \boldsymbol{U}(k)) + \boldsymbol{M}(k+1)\right], \tag{1.2a}$$

$$\boldsymbol{U}(k+1) = \boldsymbol{U}(k) + b(k)\left[\boldsymbol{G}(\boldsymbol{X}(k), \boldsymbol{U}(k)) + \boldsymbol{N}(k+1)\right], \tag{1.2b}$$

where $a(k)$ and $b(k)$ are real sequences, $\boldsymbol{R}(\cdot,\cdot)$ and $\boldsymbol{G}(\cdot,\cdot)$ are deterministic vector valued functions, and $\boldsymbol{M}(k+1)$ and $\boldsymbol{N}(k+1)$ are random noise terms. Thus, the terms in square brackets in (1.2a,b) are SA update directions for $\boldsymbol{X}(k)$ and $\boldsymbol{U}(k)$. By letting $\theta = [\boldsymbol{X}^\top, \boldsymbol{U}^\top]^\top$ and setting $\boldsymbol{R}(\cdot,\cdot) = \boldsymbol{G}(\cdot,\cdot)$ it may appear that any strictly alternating cyclic SA algorithm with disjoint subvectors is a special case of (1.2a,b), but this is not true. To see that (1.2a,b) is not a cyclic recursion, note that the recursion in (1.2a,b) implies that the values of $\boldsymbol{X}(k+1)$ and $\boldsymbol{U}(k+1)$ are both obtained based on the vector $[\boldsymbol{X}(k)^\top, \boldsymbol{U}(k)^\top]^\top$. In a cyclic algorithm, however, the values of $\boldsymbol{X}(k+1)$ and $\boldsymbol{U}(k+1)$ would be obtained based on the vectors $[\boldsymbol{X}(k)^\top, \boldsymbol{U}(k)^\top]^\top$ and $[\boldsymbol{X}(k)^\top, \boldsymbol{U}(k+1)^\top]^\top$, respectively (assuming $\boldsymbol{X}(k)$ is updated first). Therefore, the pair in (1.2a,b) is not cyclic. The remainder of this section discusses the results that most closely resemble GCSA.

Tsitsiklis (1994) considers an asynchronous SA algorithm for finding a solution to $\boldsymbol{D}(\boldsymbol{x}) = \boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^p$ is a vector and $\boldsymbol{D}(\cdot)$ is a vector valued function,

using possibly outdated information. Specifically, using Tsitsiklis' notation, the ith entry of $\boldsymbol{x}$, denoted by x_i, is updated according to the following recursion:

$$x_i(t+1) = x_i(t) - \alpha_i(t)[D_i(\boldsymbol{x}^i(t)) - x_i(t) + w_i(t)] \quad \text{for } t \in T^i,$$
$$x_i(t+1) = x_i(t) \quad \text{for } t \notin T^i, \tag{1.3}$$

where T^i is a random set of (nonnegative integer) times at which x_i is updated, $\alpha_i(t) \in [0,1]$, $w_i(t)$ is a mean-zero noise term, D_i denotes the ith entry of $\boldsymbol{D}$, and $\boldsymbol{x}^i(t)$ is a vector of (possibly) outdated components of $\boldsymbol{x}$, that is $\boldsymbol{x}^i = [x_1(\tau_1^i(t)), \ldots, x_p(\tau_p^i(t))]^\top$, where $\tau_j^i(t) \leq t$ is a nonnegative integer. If no information is outdated then $\boldsymbol{x}^i(t) = \boldsymbol{x}(t)$. It is in the case of delayed information that the resemblance to cyclic optimization becomes apparent. To see this, consider the special case of (1.3) where p is even, T^i is the set of nonnegative odd numbers for $i \leq p/2$, T^i is the set of even numbers for $i > p/2$, $\boldsymbol{x}^i(t) = \boldsymbol{x}(t)$ for $i \leq p/2$, and $\boldsymbol{x}^i(t) = [x_1(t+1), \ldots, x_{p/2}(t+1), x_{p/2+1}(t), \ldots, x_p(t)]^\top$ for $i > p/2$. The resulting algorithm is a special case of cyclic SA in which two subvectors of $\boldsymbol{x}(t)$ are updated in a strictly alternating manner.

One convenient property of (1.3) is that it may be implemented in an asynchronous manner, that is the entries of $\boldsymbol{x}(t)$ that are updated at time t can each be updating using outdated parameter vectors collected at possibly different times. The GCSA algorithm, in contrast, is synchronous in nature since the

entries of the parameter vector that are updated at any given time must all be updated using the latest value of the parameter vector. Despite the advantage that an asynchronous algorithm presents over a synchronous algorithm, there are a few reasons why the theory regarding the convergence of (1.3) does not apply to GCSA as is discussed next.

One of the conditions that Tsitsiklis (1994) imposes for the convergence of (1.3) requires the second moment of $w_i(t)$ to be bounded above by a function that depends on the magnitude of $\boldsymbol{x}(t)$ (see Assumption 2.e in Tsitsiklis 1992). When $\boldsymbol{x}(t)$ converges to some finite vector w.p.1 (e.g., to the solution of $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x}$), Tsitsiklis' (1994) assumption requires $w_i(t)$ to have bounded variance w.p.1. However, because (1.3) can be thought of as a special case of (1.1):

$$\begin{aligned}\text{Update direction in first line of (1.3)} &= D_i(\boldsymbol{x}^i(t)) - x_i(t) + w_i(t) \\ &= \underbrace{D_i(\boldsymbol{x}(t)) - x_i(t)}_{i\text{th entry of } \boldsymbol{f} \text{ term in (1.1)}} + \underbrace{D_i(\boldsymbol{x}^i(t)) - D_i(\boldsymbol{x}(t)) + w_i(t)}_{i\text{th entry of } \beta + \xi \text{ term in (1.1)}}, \end{aligned} \tag{1.4}$$

requiring $w_i(t)$ to have bounded second moment is equivalent to requiring the diagonal entries of the second moment matrix of $\beta_k(\hat{\theta}_k) + \xi_k(\hat{\theta}_k)$ in (1.1) to have bounded magnitude, an assumption that is incompatible with many SA algorithms (such as the simultaneous perturbation SA algorithm) for which the variance of the noise can increase with time. In this dissertation we allow the variance of the noise to increase as the iteration number increases in

order to accommodate a more general class of SA update directions. Another observation we make is that the vector $\beta_k(\hat{\theta}_k) + \xi_k(\hat{\theta}_k)$ from (1.1) has a very specific form in (1.4). In the GCSA algorithm, $\beta_k(\hat{\theta}_k) + \xi_k(\hat{\theta}_k)$ term is allowed to have a form that is more appropriate for general SA procedures. In conclusion, the assumptions imposed by Tsitsiklis (1994) imply that the class of algorithms studied in Tsitsiklis (1994) intersects the class of algorithms that fit into the GCSA framework although neither class of algorithms contains the other. Moreover, Tsitsiklis (1994) does not discuss rate of convergence or provide results on asymptotic normality, both contributions of this dissertation.

Another algorithm closely related to GCSA appears in Borkar (1998). Here, the author investigates the asymptotic behavior of a distributed, asynchronous SA scheme in terms of a limiting differential equation. One important assumption made by Borkar (1998) is that the update direction for the ith entry of the parameter vector is an unbiased estimate of the ith entry of $f(\theta)$, the function whose root is to be found (see equation 2.9 in Borkar 1998). This assumption is usually only valid for certain SA algorithms (e.g., the stochastic gradient algorithm) or deterministic optimization problems. The theory in this dissertation does not require the availability of unbiased estimates of the entries of $f(\theta)$. Another assumption made by Borkar (1998) requires $f(\theta)$ to be Lipschitz continuous. In our setting (which focuses on stochastic optimization via root-finding so that $f(\theta) = g(\theta)$) this would be equivalent to requiring the gradient

of $L(\theta)$ to be Lipschitz continuous. We do not make such an assumption in the theory for convergence w.p.1 of the GCSA iterates. Additionally, Borkar (1998) does not discuss rate of convergence or provide results on asymptotic normality, both contributions of this dissertation.

In this dissertation, the theory behind the GCSA algorithms allows the noisy update directions to be biased estimates of $\boldsymbol{f}(\theta)$ and allows the noise term in (1.1) (i.e., $\xi_k(\hat{\theta}_k)$) to have a second moment matrix whose entries can increase in magnitude as a function of the iteration number. Algorithms that are related to GCSA but require the availability of unbiased estimates of $\boldsymbol{f}(\theta)$ can be found in the papers by Borkar and Meyn (2000), Aboundai et al. (2002), Bhatnagar (2011), Bianchi and Jakubowicz (2013), and Singh et al. (2014) (technically, the estimate for $\boldsymbol{f}(\theta)$ in this last reference is and unbiased estimate of a vector that has the same roots as $\boldsymbol{f}(\theta)$). Algorithms that are related to GCSA but require the term $\xi_k(\hat{\theta}_k)$ in (1.1) to have bounded variance can be found in the papers by Tsitsiklis (1984), Tsitsiklis et al. (1986), Solodov and Zavriev (1998), Ram et al. (2009b), Nedić and Bertsekas (2010), Ram et al. (2010), Xu and Yin (2015), and in Exercise 1.7.1 on p. 34 in Benveniste et al. (1990). The distributed algorithms in Chapter 12 of Kushner and Yin (1997) are also closely related to GCSA. However, the results in said chapter require the sequence of noisy update directions (i.e., the sequence $\boldsymbol{Y}_k(\hat{\theta}_k)$) to be uniformly integrable (see Kushner and Yin 1997, Chapter 12, Assumptions A3.1

and A3.1′), an assumption that is too strong for some SA algorithms such as finite difference SA where the noise in the update direction generally makes the $\boldsymbol{Y}_k(\hat{\theta}_k)$ vectors not uniformly integrable. This dissertation does not make the assumption of uniform integrability of the noisy update directions. Lastly, a few results related to GCSA concerned with optimizing convex functions can be found in the papers by Ram et al. (2009a), Necoara (2013), and Necoara and Petrascu (2014) (the theory in this dissertation does not assume convexity).

1.3 This Work's Contribution

This work's main contribution can be summarized by the following points:

1. We derive conditions for the convergence with probability one of the GCSA algorithm to a root of the gradient of $L(\theta)$. The main convergence result is stated in Theorem 2 of Section 4.3. A few corollaries based on special cases of GCSA are derived in the same section. Numerical results supporting the theory on convergence are provided in Section 7.1.

2. We provide a generalization to Theorem 2.2 in Fabian (1968) (see Theorem 4) that allows us to show asymptotic normality for a special case of GCSA. Appendix A explains how the generalization to Fabian's theorem also extends the theorem's result to include a broader range of SA algorithms of practical interest including certain second order SA and

root-finding SA algorithms.

3. We show the asymptotic normality of the normalized iterates of a special case of GCSA in which the subvector to update is selected according to a deterministic pattern (see Theorem 5). The result on asymptotic normality helps us define the asymptotic rate of convergence for the iterates of this special case of GCSA. Numerical results supporting the theory on asymptotic normality are provided in Section 7.2.

4. We discuss the importance of defining the cost of implementation when comparing the performances of two optimization algorithms. When cost is a measure of the number of basic arithmetic computations required, we discuss the type of arithmetic operations that can result in a significant difference between the per-iteration cost of implementing an algorithm and the per-iteration cost of implementing its cyclic counterpart.

5. We provide an analytical estimate for the asymptotic efficiency of a special case of GCSA relative to the efficiency of its non-cyclic counterpart after taking into consideration the per-iteration costs of implementation. Here, efficiency is defined in terms of the mean-squared estimation errors. We show how the expression for asymptotic relative efficiency implies that either algorithm (cyclic or non-cyclic) can be more efficient. Numerical experiments computing the relative efficiency between cyclic and non-cyclic

algorithms under different definitions of cost are provided in Section 7.3.

6. In Chapter 8 we apply the cyclic SA approach to a multi-agent optimization problem for tracking and surveillance.

7. This dissertation also contains two appendices. Appendix A contains a more detailed proof of Theorem 4, the theorem in Chapter 5 that generalizes Theorem 2.2 in Fabian (1968), and discusses a few applications of the generalized theorem. Appendix 5 considers the problem of determining the presence and location of a static object within an area of interest by combining information from multiple sensors using a maximum-likelihood-based approach.

Two noteworthy assumptions made throughout this dissertation are that the loss function is differentiable and that the optimization problem is unconstrained. Future work could focus on investigating the constrained- and non-differentiable settings.

1.4 Overview of Contents

The main part of this work (Chapters 2–8) is organized as follows. First, Chapter 2 reviews the general stochastic optimization setting as well as the idea behind SA algorithms for stochastic optimization. Chapter 3 introduces the cyclic seesaw SA algorithm and the generalized cyclic SA algorithm (GCSA)

that is the focus of this work. Chapter 4 focuses on deriving conditions for the convergence with probability one of the GCSA algorithm's iterates to a root of the gradient of the function to minimize. Chapter 5 generalizes a well-known result in the SA literature (Fabian 1968, Theorem 2.2) and uses the resulting generalization to prove asymptotic normality of the scaled iterates for a special case of GCSA. Chapter 6 is concerned with computing the asymptotic efficiency (defined as the asymptotic mean-squared-error) of a special case of GCSA relative to that of its non-cyclic counterpart. The chapter begins by discussing the importance of defining the cost of implementation before attempting to compare any two algorithms. The chapter goes on to provide a comparison of the cost of implementing an SA algorithm in a cyclic manner versus the cost if implemented in a non-cyclic manner; a few definitions of cost are considered. Using the asymptotic normality result from Chapter 5, Chapter 6 computes the asymptotic relative efficiency between a special case of GCSA and its non-cyclic counterpart. Chapter 7 contains numerical results that illustrate the theory of Chapters 4–6. Chapter 8 applies the cyclic approach to a multi-agent optimization problem where the loss function is time-varying and corrupted by noise. While the theory from Chapters 4 and 5 (which is concerned with the optimization of a loss function that is not time-varying) is not fully applicable to this multi-agent problem (the main condition in Chapters 4–5 not satisfied is the decaying gain sequence of SA; as a tracking problem the gains are not

allowed to decay to zero), the purpose of the numerical example in this chapter is threefold. First, it addresses an important area surveillance and tracking problem. Second, it studies the performance of a cyclic implementation when the conditions from Chapters 4 and 5 do not fully hold. Third, it serves to motivate directions for future work.

Following Chapter 8 this work is organized as follows. Appendix A contains a more detailed proof of Theorem 4, a theorem in Chapter 5 that generalizes Theorem 2.2 in Fabian (1968). Appendix A also discusses how the generalization makes Theorem 2.2 in Fabian (1968) applicable to a broader range of SA algorithms that extend beyond cyclic SA. In contrast to the content of Chapters 1–8 and Appendix A, Appendix B is concerned with a topic that is unrelated to SA: it considers the problem of determining the presence and location of a static object within an area of interest by combining information from multiple sensors. A simple maximum-likelihood-based approach is investigated. Lastly, a list of frequently used notation is included at the end of this dissertation.

Chapter 2

Preliminaries

This chapter lays the groundwork for our study of stochastic optimization based on nonlinear root-finding stochastic approximation (SA) algorithms. Section 2.1 formally introduces the general stochastic optimization setting and briefly reviews the relationship between root-finding and optimization. Section 2.2 describes the basic form of SA algorithms for nonlinear root-finding. Lastly, Sections 2.3 and 2.4 give three examples of well-known SA algorithms: stochastic gradient, finite difference SA, and simultaneous perturbation SA.

2.1 Stochastic Optimization

The idea behind *unconstrained optimization* problems is the minimization of a real-valued loss function $L(\theta)$ with respect to a parameter vector θ. When $L(\theta)$ is a smooth function, its gradient is denoted by $\boldsymbol{g}(\theta) \equiv \partial L(\theta)/\partial\theta$ and,

in this case, the optimization problem can be reformulated as a root-finding problem where one attempts to solve $\boldsymbol{g}(\theta) = 0$ (this equation is referred to as the *gradient equation* and any solution is referred to as a *root of the gradient*). Here, the usual caveat applies: the set of all roots of the gradient may contain vectors other than global minimizers of $L(\theta)$. Still, regardless of this caveat, the close relationship between minimization and root-finding implies many optimization algorithms rely on being able to evaluate either $L(\theta)$ or $\boldsymbol{g}(\theta)$. Take the steepest descent algorithm (e.g., Nocedal and Wright 2006, Chapter 2), for example. This algorithm is an iterative line-search algorithm in which an estimate, $\hat{\theta}_k$, for a solution to the gradient equation is obtained during the kth iteration according to the recursion:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{g}(\hat{\theta}_k), \tag{2.1}$$

where a_k is a strictly-positive scalar often referred to as the *gain sequence* (one valid choice for the gain sequence in the steepest descent algorithm is setting a_k equal to a constant that does not depend on k). Under certain assumptions, $\hat{\theta}_k$ converges to θ^*, a root of the gradient. It is apparent from (2.1) that being able to evaluate $\boldsymbol{g}(\theta)$ at $\{\hat{\theta}_k\}_{k \geq 0}$ is essential to the implementation of the steepest-descent algorithm. Similarly, many optimization algorithms rely on being able to evaluate $L(\theta)$. Simple examples of this type of algorithm are the random

search algorithms in Section 2.2 of Spall (2003).

In practice, it is often the case that evaluating the loss function or its gradient is difficult. As a simple example, consider a setting where there exists a complex stochastic model whose output depends on a set of parameters denoted by θ. Furthermore, suppose one wishes to find the value of θ that minimizes the expected output of the model. Here, finding a closed form expression for $L(\theta)$ would require computing the expected output for each θ, which might be infeasible due to the complexity of the model. When dealing with physical processes in the optimization process, rather than mathematical models, computing the expected value of the measurement/output for any given θ could easily be impossible (rather than simply infeasible) since physical processes are often governed by rules unknown to the observer. While traditional optimization techniques cannot be implemented when neither $L(\theta)$ nor $\boldsymbol{g}(\theta)$ are known, stochastic optimization algorithms can use noisy measurements of either the loss function or its gradient in the minimization process. Hereafter, the term "*stochastic optimization*" will be used to refer to optimization problems where there is random noise in the measurements of $L(\theta)$ or $\boldsymbol{g}(\theta)$, or where there is a random choice made in the search direction as the algorithm iterates towards a solution. The term "*deterministic optimization*" will be used to refer to classical optimization problems, like steepest descent or Newton's method, where the minimization process is entirely deterministic.

As mentioned in the previous paragraph, one type of stochastic optimization problem pertains to the case where one wishes to minimize a function $L(\theta)$ when only noisy measurements of this function are available. In other words, it is assumed that the loss function is unknown but that it is possible to obtain measurements of a random variable $Q(\theta, \boldsymbol{V})$ such that:

$$Q(\theta, \boldsymbol{V}) = L(\theta) + \varepsilon(\theta, \boldsymbol{V}), \tag{2.2}$$

where $\boldsymbol{V}$ denotes a multivariate random variable and $\varepsilon(\theta, \boldsymbol{V})$ is a noise term. The term $\varepsilon(\theta, \boldsymbol{V})$ can then be interpreted as the error in measuring the function to minimize. In the special case where the expected value of $\varepsilon(\theta, \boldsymbol{V})$ at θ is equal to zero, a consequence of the law of large numbers is that it is possible to approximate $L(\theta)$ at any given θ by averaging several independent and identically distributed (i.i.d.) measurements of $Q(\theta, \boldsymbol{V})$. In theory, one could average several i.i.d. noisy loss function measurements to obtain $\bar{L}(\theta_1), \ldots, \bar{L}(\theta_N)$, a set of estimates for $L(\theta)$ at the points $\theta = \theta_1, \ldots, \theta_N$ (selected from the domain of $L(\theta)$ via some deterministic or random scheme). Then, an approximation to $L(\theta)$ for all θ, denoted by $\bar{L}(\theta)$, could be obtained by interpolating the values of $\bar{L}(\theta_i)$ (obtaining this interpolation may be a nontrivial task). Afterwards, one could attempt to minimize $\bar{L}(\theta)$ using deterministic optimization algorithms with the hope that the minimizer of $\bar{L}(\theta)$ is close to the minimizer or $L(\theta)$ (the

response surface methodology strategy, introduced by Box and Wilson in 1951, is a more sophisticated variant of this approach). Such an approach, however, is not always practical since obtaining a good approximation to the loss function (an approximation to the loss function would be considered "good" if its minimizer is close to the minimizer of $L(\theta)$) could require a prohibitive number of noisy function measurements. The following section describes the stochastic approximation setting, a general framework for stochastic nonlinear root-finding that is more appropriate for minimizing (2.2) than the deterministic optimization approach discussed above.

2.2 Stochastic Approximation

There exist many stochastic optimization algorithms. Random search, genetic algorithms, simulated annealing, stochastic gradient, and simultaneous perturbation SA, are a few examples. This work will focus on *stochastic approximation* (SA) algorithms for stochastic optimization via nonlinear root-finding. SA algorithms are closely related to line-search methods and can be used for stochastic optimization. This section formally defines SA and presents three important examples of SA algorithms: the *stochastic gradient* (SG) form of SA, *finite difference* SA (FDSA), and *simultaneous perturbation* SA (SPSA). Throughout the remainder of this work it is assumed that $\theta = [\tau_1, \ldots, \tau_p]^\top \in \mathbb{R}^p$

and θ^* will denote a solution to $\boldsymbol{g}(\theta) = 0$.

The basic SA algorithm for nonlinear root-finding is known as the Robbins–Monro algorithm (Robbins and Monro 1951). Given a vector-valued function $\boldsymbol{f}(\theta)$, the Robbins–Monro algorithm attempts to find a solution to $\boldsymbol{f}(\theta) = 0$ in an iterative manner using the following recursion:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{Y}_k(\hat{\theta}_k), \tag{2.3}$$

where $\boldsymbol{Y}_k(\hat{\theta}_k)$ is a vector-valued random variable representing a noisy observation of $\boldsymbol{f}(\hat{\theta}_k)$, and $a_k > 0$ is the gain sequence (step size). Unlike most deterministic optimization algorithms, the gain sequence in (2.3) typically satisfies $a_k \to 0$. This dissertation is concerned with the special case where $\boldsymbol{f}(\theta) = \boldsymbol{g}(\theta)$. In this context, $\boldsymbol{Y}_k(\hat{\theta}_k)$ denotes a noisy estimate of the gradient and it is common to replace the notation $\boldsymbol{Y}_k(\hat{\theta}_k)$ in (2.3) with $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$. Therefore, we write:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\theta}_k), \tag{2.4}$$

where $\hat{\boldsymbol{g}}(\hat{\theta}_k)$ is a noisy gradient measurement.

Because $\hat{\boldsymbol{g}}(\hat{\theta}_k)$ can be thought of as an estimate of $\boldsymbol{g}(\hat{\theta}_k)$, the theory of SA for stochastic optimization often relies on rewriting the vector $\hat{\boldsymbol{g}}(\hat{\theta}_k)$ as:

$$\hat{\boldsymbol{g}}_k(\hat{\theta}_k) = \boldsymbol{g}(\hat{\theta}_k) + \beta_k(\hat{\theta}_k) + \xi_k(\hat{\theta}_k), \tag{2.5}$$

a special case of (1.1), where $\beta_k(\hat{\theta}_k) = E\big[\hat{\boldsymbol{g}}_k(\hat{\theta}_k) - \boldsymbol{g}(\hat{\theta}_k)\big|\mathcal{F}_k\big]$, $\xi_k(\hat{\theta}_k) = \hat{\boldsymbol{g}}_k(\hat{\theta}_k) - E\big[\hat{\boldsymbol{g}}_k(\hat{\theta}_k)\big|\mathcal{F}_k\big]$, $\mathcal{F}_k$ is some representation of the history of the process (the precise definition of $\mathcal{F}_k$ may vary from algorithm to algorithm), and $E[\mathcal{X}]$ represents the expected value of the random variable $\mathcal{X}$. One common choice of $\mathcal{F}_k$ is $\mathcal{F}_k = \hat{\theta}_0, \ldots, \hat{\theta}_k$. In this case, $\beta_k(\hat{\theta}_k)$ represents the *bias* of $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$ (e.g., Bickel and Doksum 2007) as an estimator of $\boldsymbol{g}(\hat{\theta}_k)$ and the vector $\xi_k(\hat{\theta}_k)$ is often referred to as the *noise* term. In the special case where $E\big[\hat{\boldsymbol{g}}_k(\hat{\theta}_k)\big|\hat{\theta}_k\big] = \boldsymbol{g}(\hat{\theta}_k)$, $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$ is said to be an unbiased estimate of the gradient at $\hat{\theta}_k$. It is important to mention that the decomposition in (2.5) is used only for theoretical purposes and, in practice, the bias and noise terms are never computed. Sections 2.3 and 2.4 discuss special cases of (2.4) that differ in the way $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$ is computed.

2.3 Stochastic Gradient Form of SA

The *stochastic gradient* (SG) algorithm for stochastic approximation (e.g., Spall 2003, Chapter 5) is a special case of the Robbins-Monro algorithm that requires the availability of a vector $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$ such that $E\big[\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)\big|\hat{\theta}_k\big] = \boldsymbol{g}(\hat{\theta}_k)$ (i.e., the noisy gradient measurement must be an unbiased measurement of the gradient at $\hat{\theta}_k$). The recursion defining $\hat{\theta}_k$ in the SG algorithm is as follows:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k),$$

a special case of (2.4). Because $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$ is an unbiased estimate of $\boldsymbol{g}(\hat{\theta}_k)$ we write $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k) = \boldsymbol{g}(\hat{\theta}_k) + \boldsymbol{\xi}_k^{\mathrm{SG}}(\hat{\theta}_k)$, where $E[\boldsymbol{\xi}_k^{\mathrm{SG}}(\hat{\theta}_k)|\mathcal{F}_k] = 0$. In the area of simulation-based optimization, two popular algorithms that are special cases of the SG algorithm are the pure infinitesimal perturbation analysis (IPA) algorithm and the pure likelihood ratio function algorithm (Spall 2003, p. 418). For both these algorithms, the random vector $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$ can sometimes be obtained by assuming that is is possible to differentiate $Q(\theta, \boldsymbol{V})$ with respect to θ (e.g., Spall 2003, p. 134). This approach is formally discussed next.

First note that (2.2) implies $E[Q(\theta, \boldsymbol{V})|\theta] = L(\theta) + E[\varepsilon(\theta, \boldsymbol{V})|\theta]$ so that:

$$\frac{\partial}{\partial \theta} E[Q(\theta, \boldsymbol{V})|\theta] = \boldsymbol{g}(\theta) + \frac{\partial E[\varepsilon(\theta, \boldsymbol{V})|\theta]}{\partial \theta}. \tag{2.6}$$

If $E[\varepsilon(\theta, \boldsymbol{V})|\theta]$ does not depend on θ (e.g., if ε is i.i.d. noise) then (2.6) implies:

$$\boldsymbol{g}(\theta) = \frac{\partial}{\partial \theta} E[Q(\theta, \boldsymbol{V})|\theta] = \frac{\partial}{\partial \theta} \int_{\Omega_{\boldsymbol{V}}} Q(\theta, \boldsymbol{v}) \, dP, \tag{2.7}$$

where P denotes a probability measure and $\Omega_{\boldsymbol{V}}$ is the domain of $\boldsymbol{V}$. Consider the case where $\boldsymbol{V}$ is a continuous random variable with probability density function (pdf) $p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)$ that may depend on θ. Then, (2.7) becomes:

$$\boldsymbol{g}(\theta) = \frac{\partial}{\partial \theta} \int_{\Omega_{\boldsymbol{V}}} Q(\theta, \boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta) \, d\boldsymbol{v}.$$

If the interchange of differentiation and integration is justified we obtain:

$$\boldsymbol{g}(\theta) = \int_{\Omega_{\boldsymbol{V}}} \left[\frac{\partial Q(\theta, \boldsymbol{v})}{\partial \theta} + Q(\theta, \boldsymbol{v}) \frac{\partial \log(p_{\boldsymbol{V}}(\boldsymbol{v}|\theta))}{\partial \theta} \right] p_{\boldsymbol{V}}(\boldsymbol{v}|\theta) \, d\boldsymbol{v}. \tag{2.8}$$

From (2.8) we see that

$$\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k) = \frac{\partial Q(\theta, \boldsymbol{V})}{\partial \theta}\Big|_{\theta=\hat{\theta}_k} + Q(\hat{\theta}_k, \boldsymbol{V}) \frac{\partial \log[p_{\boldsymbol{V}}(\boldsymbol{V}|\theta)]}{\partial \theta}\Big|_{\theta=\hat{\theta}_k}, \tag{2.9}$$

is an unbiased measurement of $\boldsymbol{g}(\hat{\theta}_k)$ by construction. The following condition is sufficient for the validity of the interchange of differentiation and integration used in (2.8).

Theorem 1 (Interchange of Differentiation and Integration). (Spall 2003, Theorem 15.1). Assume $Q(\theta, \boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)$ and $\partial[Q(\theta, \boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)]/\partial\theta$ are continuous on $\mathbb{R}^p \times \Omega_{\boldsymbol{V}}$. Suppose there exist two nonnegative integrable functions $q_0(\boldsymbol{v})$ and $q_1(\boldsymbol{v})$ such that for all pairs $(\theta, \boldsymbol{v}) \in \mathbb{R}^p \times \Omega_{\boldsymbol{V}}$:

$$|Q(\theta, \boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)| \leq q_0(\boldsymbol{v}); \quad \left\| \frac{\partial [Q(\theta, \boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)]}{\partial \theta} \right\| \leq q_1(\boldsymbol{v}).$$

Then:

$$\boldsymbol{g}(\theta) = \frac{\partial}{\partial \theta} \int_{\Omega_{\boldsymbol{V}}} Q(\theta, \boldsymbol{V}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta) \, d\boldsymbol{v} = \int_{\Omega_{\boldsymbol{V}}} \frac{\partial}{\partial \theta} \left[Q(\theta, \boldsymbol{V}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta) \right] \, d\boldsymbol{v}.$$

When $\boldsymbol{V}$ is a discrete random variable taking the values $1, \ldots, N$, a relationship analogous to (2.8) can be obtained. To see this, let $p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)$ denote the probability mass function (pmf) of $\boldsymbol{V}$. In this case,

$$\boldsymbol{g}(\theta) = \frac{\partial}{\partial \theta} \sum_{\boldsymbol{v}=1}^{N} Q(\theta, \boldsymbol{v}) p_{\boldsymbol{V}}(\boldsymbol{v}|\theta). \tag{2.10}$$

If $N < \infty$, the interchange of change of summation and differentiation in (2.10) is justified (provided the required derivatives exist). This may not be the case, however, when $N = \infty$ (the sum of the gradients of the individual summands may fail to converge or may converge to something other than $\boldsymbol{g}(\theta)$). When $N = \infty$, sufficient conditions for the interchange of differentiation and summation can easily be obtained using Theorem 7.17 in Rudin (1976) (although this theorem pertains only to the case where θ is real, the result of the theorem is easily generalized to the multi-dimensional case by applying the theorem to each entry of the gradient vectors). When the exchange of differentiation and summation in (2.10) is valid,

$$\boldsymbol{g}(\theta) = \sum_{\boldsymbol{v}=1}^{N} \left[\frac{\partial Q(\theta, \boldsymbol{v})}{\partial \theta} + Q(\theta, \boldsymbol{v}) \frac{\partial \log[p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)]}{\partial \theta} \right] p_{\boldsymbol{V}}(\boldsymbol{v}|\theta).$$

From this, we deduce that (2.9) once again represents an unbiased measurement of the gradient.

While (2.9) gives a theoretically valid expression for $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$, it requires

information that may not be available. For example, if $p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)$ is unknown we would not be able to compute $\partial \log [p_{\boldsymbol{V}}(\boldsymbol{V}|\theta)]/\partial\theta$. However, if $p_{\boldsymbol{V}}(\boldsymbol{v}|\theta)$ is independent of θ (a common assumption) then (2.9) simplifies to:

$$\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k) = \frac{\partial Q(\theta, \boldsymbol{V})}{\partial \theta}\Big|_{\theta=\hat{\theta}_k}. \tag{2.11}$$

Then, $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$ could be obtained through direct measurement of $\partial Q(\theta, \boldsymbol{V})/\partial\theta$ evaluated at $\theta = \hat{\theta}_k$. In the area of simulation-based optimization, the Robbins–Monro algorithm with $\hat{\boldsymbol{g}}_k(\hat{\theta}_k) = \hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$ where $\hat{\boldsymbol{g}}_k^{\mathrm{SG}}(\hat{\theta}_k)$ is as in (2.11) is commonly referred to as the pure IPA algorithm.

2.4 Simultaneous Perturbation SA

SA is a powerful tool for stochastic optimization. Section 2.3 discussed the SG algorithm which used a noisy but unbiased measurement of $\boldsymbol{g}(\hat{\theta}_k)$ to update $\hat{\theta}_k$. Section 2.3 also gave conditions under which a direct measurement of $\partial Q(\theta, \boldsymbol{V})/\partial\theta$ could be used as the noisy unbiased measurement of the gradient. Direct measurement of $\partial Q(\theta, \boldsymbol{V})/\partial\theta$ is certainly a feasible approach in some applications (e.g., Widrow and Stearns 1985). However, measuring $\partial Q(\theta, \boldsymbol{V})/\partial\theta$ is not always possible and is particularly a problem in black-box settings where the form of $Q(\theta, \boldsymbol{V})$ is unknown. With this motivation, several SA approximate $\boldsymbol{g}(\theta)$ only using noisy measurements of $Q(\theta, \boldsymbol{V})$ and, for this

reason, these SA algorithms are often said to be *gradient-free*.

Let us briefly discuss the oldest gradient-free SA method: *finite difference stochastic approximation* (FDSA). See, for example, Dennis and Schnabel (1989). FDSA requires measuring $Q(\theta, \boldsymbol{V})$ at different values of θ. Specifically, $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{\boldsymbol{g}}_k^{\mathrm{FD}}(\hat{\theta}_k)$, where the random vector $\hat{\boldsymbol{g}}_k^{\mathrm{FD}}(\hat{\theta}_k)$ is defined as follows:

$$\hat{\boldsymbol{g}}^{\mathrm{FD}}(\hat{\theta}_k) \equiv \begin{bmatrix} \dfrac{Q(\hat{\theta}_k + c_k \boldsymbol{e}_1, \boldsymbol{V}^{1+}) - Q(\hat{\theta}_k - c_k \boldsymbol{e}_1, \boldsymbol{V}^{1-})}{2c_k} \\ \vdots \\ \dfrac{Q(\hat{\theta}_k + c_k \boldsymbol{e}_p, \boldsymbol{V}^{p+}) - Q(\hat{\theta}_k - c_k \boldsymbol{e}_p, \boldsymbol{V}^{p-})}{2c_k} \end{bmatrix}, \tag{2.12}$$

where $\boldsymbol{e}_i$ denotes the ith standard-basis vector in $\mathbb{R}^p$, $c_k > 0$ satisfies $c_k \to 0$, and where $\boldsymbol{V}^{1+}, \ldots, \boldsymbol{V}^{p+}, \boldsymbol{V}^{1-}, \ldots, \boldsymbol{V}^{p-}$ denote $2p$ different realizations of $\boldsymbol{V}$. The update direction in (2.12) is referred to as the two-sided FDSA update direction and requires a total of $2p$ noisy loss function measurements at each iteration (recall that p is the dimension of the parameter space). When p is large and if obtaining noisy loss function measurements is costly, requiring $2p$ noisy loss function measurements per-iteration could be prohibitive (this is also an issue for a one-sided version of the FDSA algorithm requiring $p + 1$ noisy loss measurements per-iteration).

The *simultaneous perturbation stochastic approximation* (SPSA) algorithm (e.g., Spall 1992; Bhatnagar et al. 2013) is similar to FDSA but requires only two noisy loss function measurements per iteration (independently of p). In

SPSA, the gradient approximation in (2.12) is replaced with the random vector:

$$\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) \equiv \begin{bmatrix} \dfrac{Q(\hat{\theta}_k + c_k\boldsymbol{\Delta}_k, \boldsymbol{V}^+) - Q(\hat{\theta}_k - c_k\boldsymbol{\Delta}_k, \boldsymbol{V}^-)}{2c_k\Delta_{k1}} \\ \vdots \\ \dfrac{Q(\hat{\theta}_k + c_k\boldsymbol{\Delta}_k, \boldsymbol{V}^+) - Q(\hat{\theta}_k - c_k\boldsymbol{\Delta}_k, \boldsymbol{V}^-)}{2c_k\Delta_{kp}} \end{bmatrix}, \tag{2.13}$$

where $\boldsymbol{\Delta}_k = [\Delta_{k1}, \ldots, \Delta_{kp}]^\top$ is a random vector, $c_k > 0$ is a sequence satisfying $c_k \to 0$, and $\boldsymbol{V}^+$ and $\boldsymbol{V}^-$ are two different realizations of $\boldsymbol{V}$. The SPSA recursion governing $\{\hat{\theta}_k\}_{k\geq 0}$ is then $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)$. By defining $\mathcal{F}_k \equiv \{\hat{\theta}_0, \ldots, \hat{\theta}_k, \boldsymbol{\Delta}_0, \ldots, \boldsymbol{\Delta}_{k-1}\}$, $\beta_k^{\mathrm{SP}}(\hat{\theta}_k) \equiv E[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) - \boldsymbol{g}(\hat{\theta}_k)|\mathcal{F}_k]$, and $\xi_k^{\mathrm{SP}}(\hat{\theta}_k) \equiv \hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) - \boldsymbol{g}(\hat{\theta}_k) - \beta_k^{\mathrm{SP}}(\hat{\theta}_k)$, the connection between the SPSA algorithm and (2.4)–(2.5) becomes apparent.

Let us briefly discuss an important property of $\beta_k^{\mathrm{SP}}(\hat{\theta}_k)$ and $\xi_k^{\mathrm{SP}}(\hat{\theta}_k)$, the bias and noise terms associated with the SPSA algorithm. In (2.13), the vector $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)$ was obtained by adding a perturbation to $\hat{\theta}_k$ in two opposite directions and collecting a noisy loss measurement at these new locations. Spall (1992) derives conditions under which the bias of the gradient estimate decreases w.p.1 as c_k decreases. Spall (1992) also derives conditions under which $\beta_k(\hat{\theta}_k^{\mathrm{SP}}) = O(c_k^2)$ w.p.1, where $O(\cdot)$ is the standard big-O notation. This result implies that the bias must decrease at least as fast as c_k^2 w.p.1. In other words, as the magnitude of the perturbation decreases, the expected update direction approaches the negative gradient of $L(\theta)$. In contrast, the noise term $\xi_k^{\mathrm{SP}}(\hat{\theta}_k)$

tends to grow as c_k decreases. More specifically, the magnitude of the covariance matrix of $\mathfrak{E}_k^{\mathrm{SP}}(\hat{\boldsymbol{\theta}}_k)$ often increases at least as fast as $1/c_k^2$ (e.g., p. 391 in Spall 2003). Here, having $a_k \to 0$ serves to dampen the effect of the noise's growing variance.

2.5 Concluding Remarks

This chapter formally described the stochastic optimization problem and presented the general form of SA algorithms. Through the connection between root-finding and optimization, SA algorithms can be used for stochastic optimization. Although this dissertation focuses on stochastic optimization, the results of Chapters 4–6 also apply to the general root-finding setting (where the update directions are of the form in (1.1)) to the extent that the assumptions in Chapters 4–6 remain valid. The following chapter introduces the SA-based algorithm to be the focus of this dissertation.

Chapter 3

Basic and Generalized Cyclic SA

In the basic cyclic optimization algorithm the parameter vector θ is represented in terms of two subvectors:

$$\theta = \begin{bmatrix} \theta^{[1]} \\ \theta^{[2]} \end{bmatrix} \in \mathbb{R}^p, \tag{3.1}$$

where subvector $\theta^{[1]}$ has length p' with $0 < p' < p$. In other words, if $\theta = [\tau_1, \ldots, \tau_p]^\top$ then $\theta^{[1]} = [\tau_1, \ldots, \tau_{p'}]^\top$ and $\theta^{[2]} = [\tau_{p'+1}, \ldots, \tau_p]^\top$. The idea behind cyclic optimization methods is to alternate between updating one of the subvectors while holding the other fixed, the iterative process continues until a stopping criterion has been satisfied. Since θ is divided into two subvectors we refer to this process as cyclic *seesaw* optimization due to its back-and-forth nature. In this chapter we introduce the cyclic seesaw SA algorithm, a cyclic

implementation of SA procedures for nonlinear root-finding. This chapter also introduces the generalization to cyclic seesaw SA which will be the focus of this work (see Section 3.4). One extension permitted by the generalization pertains to a version of the algorithm that is not necessarily strictly alternating (not cyclic seesaw). Here, there may be more than two subvectors possibly with shared components and it is known that each will be updated infinitely often as the iteration count grows to infinity. A special case of this extension arises, for example, when at each iteration a random variable dictates which subvector to update. Section 3.1 formally describes the cyclic seesaw SA algorithm (with two special cases given in Sections 3.2 and 3.3) and Section 3.4 describes the generalized cyclic SA algorithm. Section 3.5 contains concluding remarks.

3.1 Cyclic Seesaw SA

We first introduce the following notation: given any vector $\boldsymbol{v} \in \mathbb{R}^p$, let the vectors $\boldsymbol{v}^{(1)}$ and $\boldsymbol{v}^{(2)}$ be the result of replacing the last $p - p'$ coordinates or the first p' coordinates of $\boldsymbol{v}$ with zeros, respectively. Specifically,

$$\boldsymbol{v}^{(1)} = \begin{bmatrix} v_1 \\ \vdots \\ v_{p'} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^p, \quad \boldsymbol{v}^{(2)} = \begin{bmatrix} \mathbf{0} \\ v_{p'+1} \\ \vdots \\ v_p \end{bmatrix} \in \mathbb{R}^p, \tag{3.2}$$

where v_i denotes the ith entry of $\boldsymbol{v}$ and 0 denotes a vector of zeros. Thus, for example, $\boldsymbol{g}(\theta) = \boldsymbol{g}^{(1)}(\theta) + \boldsymbol{g}^{(2)}(\theta)$. Using the notation in (3.2), the following recursion defines cyclic seesaw SA:

$$\hat{\theta}_k^{(I)} = \hat{\theta}_k - a_k^{(1)} \hat{\boldsymbol{g}}_k^{(1)}(\hat{\theta}_k), \quad \hat{\theta}_{k+1} = \hat{\theta}_k^{(I)} - a_k^{(2)} \hat{\boldsymbol{g}}_k^{(2)}(\hat{\theta}_k^{(I)}), \tag{3.3}$$

where $a_k^{(1)}$ and $a_k^{(2)}$ are sequences of positive scalars and $\hat{\boldsymbol{g}}_k^{(1)}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}_k^{(2)}(\hat{\theta}_k^{(I)})$ denote noisy approximations of $\boldsymbol{g}^{(1)}(\hat{\theta}_k)$ and $\boldsymbol{g}^{(2)}(\hat{\theta}_k^{(I)})$, respectively. Throughout our description of the cyclic seesaw algorithm we use the superscript "(I)" to denote the intermediate step of an iteration, this is done to emphasize the seesaw nature of the algorithm. Using (2.5), (3.3) may also be rewritten as:

$$\hat{\theta}_k^{(I)} = \hat{\theta}_k - a_k^{(1)} \left[\boldsymbol{g}^{(1)}(\hat{\theta}_k) + \beta_k^{(1)}(\hat{\theta}_k) + \xi_k^{(1)}(\hat{\theta}_k) \right], \tag{3.4a}$$

$$\hat{\theta}_{k+1} = \hat{\theta}_k^{(I)} - a_k^{(2)} \left[\boldsymbol{g}^{(2)}(\hat{\theta}_k^{(I)}) + \beta_k^{(2)}(\hat{\theta}_k^{(I)}) + \xi_k^{(2)}(\hat{\theta}_k^{(I)}) \right]. \tag{3.4b}$$

The following section describes the cyclic seesaw SG algorithm, a special case of (3.4a,b) where the gradient estimates are unbiased (see Section 2.3).

3.2 Cyclic Seesaw SG

The cyclic seesaw SG algorithm produces its updates according to (3.3) by using unbiased noisy gradient measurements to obtain the update directions

$\hat{\boldsymbol{g}}_k^{(1)}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}_k^{(2)}(\hat{\theta}_k^{(I)})$. Specifically, cyclic seesaw SG recursion is as follows:

$$\hat{\theta}_k^{(I)} = \hat{\theta}_k - a_k^{(1)} \left[\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k)\right]^{(1)}, \quad \hat{\theta}_{k+1} = \hat{\theta}_k^{(I)} - a_k^{(2)} \left[\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k^{(I)})\right]^{(2)}, \tag{3.5}$$

where $\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k^{(I)})$ are the SG estimates of $\boldsymbol{g}(\theta)$ at $\hat{\theta}_k$ and $\hat{\theta}_k^{(I)}$, respectively (see Section 2.3). Because the SG estimates are unbiased (by assumption), there exist vectors $\xi_k^{\mathrm{SG}}(\hat{\theta}_k)$ and $\xi_k^{\mathrm{SG}}(\hat{\theta}_k^{(I)})$ with $E[\xi_k^{\mathrm{SG}}(\hat{\theta}_k)|\hat{\theta}_k] = 0$ and $E[\xi_k^{\mathrm{SG}}(\hat{\theta}_k^{(I)})|\hat{\theta}_k^{(I)}] = 0$ such that:

$$\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k) = \boldsymbol{g}(\hat{\theta}_k) + \xi_k^{\mathrm{SG}}(\hat{\theta}_k), \quad \hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k^{(I)}) = \boldsymbol{g}(\hat{\theta}_k^{(I)}) + \xi_k^{\mathrm{SG}}(\hat{\theta}_k^{(I)}). \tag{3.6}$$

Cyclic seesaw SG is then a special case of (3.4a,b) in which the bias terms are zero. It is important to note that the vectors $\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k^{(I)})$ in (3.6) are defined only for theoretical purposes and, in practice, (3.5) implies it is unnecessary to compute all the entries in $\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}^{\mathrm{SG}}(\hat{\theta}_k^{(I)})$ since several entries are replaced with zeros in (3.5). The following section introduces the cyclic seesaw SPSA algorithm, another special case of cyclic seesaw SA.

3.3 Cyclic Seesaw SPSA

In the SPSA algorithm, two noisy loss function measurements are used to update the vector $\hat{\theta}_k$ (see Section 2.4). To implement SPSA in a cyclic seesaw

manner, one possibility is to replace the noisy gradient estimates in (3.3) with gradient approximations of the form in (2.13). In other words, cyclic SPSA could be implemented as follows:

$$\hat{\theta}_k^{(I)} = \hat{\theta}_k - a_k^{(1)}[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)]^{(1)}, \quad \hat{\theta}_{k+1} = \hat{\theta}_k^{(I)} - a_k^{(2)}[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})]^{(2)}, \tag{3.7}$$

where $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})$ are noisy estimates of $\boldsymbol{g}(\hat{\theta}_k)$ and $\boldsymbol{g}(\hat{\theta}_k^{(I)})$, respectively, obtained using SPSA (see Section 2.4). Specifically,

$$\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) = \begin{bmatrix} \dfrac{Q(\hat{\theta}_k + c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^+) - Q(\hat{\theta}_k - c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^-)}{2c_k^{(1)}\Delta_{k1}} \\ \vdots \\ \dfrac{Q(\hat{\theta}_k + c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^+) - Q(\hat{\theta}_k - c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^-)}{2c_k^{(1)}\Delta_{kp}} \end{bmatrix}, \tag{3.8}$$

for some sequence $\{c_k^{(1)}\}_{k\geq 0}$ and random vector $\boldsymbol{\Delta}_k = [\Delta_{k1}, \ldots, \Delta_{kp}]^\top$, and

$$\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)}) = \begin{bmatrix} \dfrac{Q(\hat{\theta}_k^{(I)} + c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)+}) - Q(\hat{\theta}_k^{(I)} - c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)-})}{2c_k^{(2)}\Delta_{k1}^{(I)}} \\ \vdots \\ \dfrac{Q(\hat{\theta}_k^{(I)} + c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)+}) - Q(\hat{\theta}_k^{(I)} - c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)-})}{2c_k^{(2)}\Delta_{kp}^{(I)}} \end{bmatrix}, \tag{3.9}$$

for a sequence $\{c_k^{(2)}\}_{k\geq 0}$ (not necessarily equal to $\{c_k^{(1)}\}_{k\geq 0}$) and a random vector $\boldsymbol{\Delta}_k^{(I)} = [\Delta_{k1}^{(I)}, \ldots, \Delta_{kp}^{(I)}]^\top$, where $\boldsymbol{V}_k^+, \boldsymbol{V}_k^-, \boldsymbol{V}_k^{(I)+}, \boldsymbol{V}_k^{(I)-}$ denote four different realizations of $\boldsymbol{V}$. Note that the definitions of $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})$ given in (3.8)

and (3.9) allow adding a small perturbation to *all* elements of the vectors $\hat{\theta}_k$ and $\hat{\theta}_k^{(I)}$, respectively. An alternative definition of cyclic seesaw SPSA can be obtained by only perturbing the entries to be updated. This is discussed next.

From (3.7) it follows that there is no need to estimate the last p' entries of $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)$ or the first $p - p'$ entries of $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})$. Thus, an alternative definition of cyclic seesaw SPSA can be obtained by setting:

$$\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) = \begin{bmatrix} \dfrac{Q(\hat{\theta}_k + c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^+) - Q(\hat{\theta}_k - c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^-)}{2c_k^{(1)}\Delta_{k1}} \\ \vdots \\ \dfrac{Q(\hat{\theta}_k + c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^+) - Q(\hat{\theta}_k - c_k^{(1)}\boldsymbol{\Delta}_k, \boldsymbol{V}_k^-)}{2c_k^{(1)}\Delta_{kp'}} \\ \mathbf{0} \end{bmatrix}, \tag{3.10}$$

where $\boldsymbol{\Delta}_k = [\Delta_{k1}, \ldots, \Delta_{kp'}, 0, \ldots, 0]^\top$, and:

$$\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)}) = \begin{bmatrix} \mathbf{0} \\ \dfrac{Q(\hat{\theta}_k^{(I)} + c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)+}) - Q(\hat{\theta}_k^{(I)} - c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)-})}{2c_k^{(2)}\Delta_{k(p'+1)}^{(I)}} \\ \vdots \\ \dfrac{Q(\hat{\theta}_k^{(I)} + c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)+}) - Q(\hat{\theta}_k^{(I)} - c_k^{(2)}\boldsymbol{\Delta}_k^{(I)}, \boldsymbol{V}_k^{(I)-})}{2c_k^{(2)}\Delta_{kp}^{(I)}} \end{bmatrix}, \tag{3.11}$$

where $\boldsymbol{\Delta}_k^{(I)} = [0, \ldots, 0, \Delta_{k(p'+1)}^{(I)}, \ldots, \Delta_{kp}^{(I)}]^\top$.

Regardless of whether $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)$ and $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)}$ in (3.7) are obtained according to (3.8)–(3.9) or according to (3.10)–(3.11), the resulting algorithm is a special case

of (3.4a,b). To see this, define $\mathcal{F}_k \equiv \{\hat{\theta}_0, \ldots, \hat{\theta}_k, \boldsymbol{\Delta}_0, \ldots, \boldsymbol{\Delta}_{k-1}, \boldsymbol{\Delta}_0^{(I)}, \ldots, \boldsymbol{\Delta}_{k-1}^{(I)}\}$ and $\mathcal{F}_k^{(I)} \equiv \{\mathcal{F}_k, \hat{\theta}_k^{(I)}, \boldsymbol{\Delta}_k\}$. Then, define the bias terms in (3.4a,b) as follows:

$$\beta_k^{(1)}(\hat{\theta}_k) = E\left[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) - \boldsymbol{g}(\hat{\theta}_k)\middle|\mathcal{F}_k\right]^{(1)}, \tag{3.12a}$$

$$\beta_k^{(2)}(\hat{\theta}_k^{(I)}) = E\left[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)}) - \boldsymbol{g}^{(2)}(\hat{\theta}_k^{(I)})\middle|\mathcal{F}_k^{(I)}\right]^{(2)}, \tag{3.12b}$$

and define the noise terms in (3.4a,b) as follows:

$$\xi_k^{(1)}(\hat{\theta}_k) = \left(\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k) - E[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)\middle|\mathcal{F}_k]\right)^{(1)}, \tag{3.13a}$$

$$\xi_k^{(2)}(\hat{\theta}_k^{(I)}) = \left(\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)}) - E\left[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})\middle|\mathcal{F}_k^{(I)}\right]\right)^{(2)}. \tag{3.13b}$$

We may then write $[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k)]^{(1)} = \boldsymbol{g}^{(1)}(\hat{\theta}_k) + \beta_k^{(1)}(\hat{\theta}_k) + \xi_k^{(1)}(\hat{\theta}_k)$ and, similarly, $[\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})]^{(2)} = \boldsymbol{g}^{(2)}(\hat{\theta}_k^{(I)}) + \beta_k^{(2)}(\hat{\theta}_k^{(I)}) + \xi_k^{(2)}(\hat{\theta}_k^{(I)})$. Therefore, cyclic seesaw SPSA is a special case of (3.4a,b). The following section introduces the generalized cyclic SA algorithm, the algorithm to be the focus of this dissertation.

3.4 Generalized Cyclic SA

While the cyclic seesaw algorithm alternates between updating each of two subvectors of θ, it is easy to conceive of an algorithm that does not exhibit this strictly alternating nature. For example, suppose that the vector to update is chosen according to a Bernoulli (i.e., binary) random variable with parameter

q_k ("success" in the Bernoulli trial could mean the first subvector is updated in which case "failure" would mean the second subvector is updated). Formally,

$$\begin{aligned} &\text{Let } \mathcal{X}_k \sim \text{Bernoulli}(q_k), \\ &\text{Let } j = 1 \text{ if } \mathcal{X}_k = 1 \text{ and } j = 2 \text{ otherwise}, \\ &\hat{\theta}_{k+1} = \hat{\theta}_k - \tilde{a}_k^{(j)} \hat{\boldsymbol{g}}_k^{(j)}(\hat{\theta}_k) \end{aligned} \tag{3.14}$$

where $\tilde{a}_k^{(j)}$ is the first unused element of a sequence $\{a_i^{(j)}\}_{i \geq 0}$ with $a_i^{(j)} > 0$, q_k is a value in the interval $(0, 1)$, and where $\mathcal{X}_k \sim \text{Bernoulli}(q_k)$ implies $\mathcal{X}_k$ has a Bernoulli distribution with success probability q_k.

Let us draw attention to two important differences between (3.3) and (3.14). First, because there is no intermediate step per se in (3.14), the superscript "(I)" is avoided altogether. Second, the deterministic gain $a_k^{(j)}$ in (3.3) has been replaced by the random gain $\tilde{a}_k^{(j)}$. Since the convergence theory for SA procedures typically requires $a_k \to 0$, using $\tilde{a}_k^{(j)}$ in place of $a_k^{(j)}$ is done with the intention of ensuring that the gain sequence does not become too small too soon. Suppose, for example, that $\mathcal{X}_k = 1$ for $k = 0, \cdots, 99$ and $\mathcal{X}_{100} = 0$ so that the first time the second subvector is updated is during iteration 101. Setting

$$\hat{\theta}_{101} = \hat{\theta}_{100} - a_{100}^{(2)} \, \hat{\boldsymbol{g}}_{100}^{(2)}(\hat{\theta}_{100})$$

in this case might prove disadvantageous since $a_{100}^{(2)}$ could be very small, thus

resulting in only a very small modification to the second subvector even when $\|\hat{\boldsymbol{g}}_{100}^{(2)}(\hat{\theta}_{100})\|$ is not close to zero. This is less of an issue when using $\tilde{a}_k^{(j)}$ since $\tilde{a}_{100}^{(2)} = a_0^{(2)}$, a gain sequence value that is likely much larger than $a_{100}^{(2)}$.

As another example of how the cyclic seesaw algorithm might be generalized, the subvectors could be updated according to some predetermined pattern (cyclic seesaw is a particular case). For example, suppose that at each iteration the parameter vector is updated according to the following repeating pattern: two updates to $\theta^{[1]}$ followed by a single update to $\theta^{[2]}$ and, lastly, two updates to $\theta^{[1]}$ (see (3.1) for the definition of $\theta^{[j]}$). This algorithm could be written as:

$$\hat{\theta}_k^{(I_{1,1})} \equiv \hat{\theta}_k - a_k^{(1)}\, \hat{\boldsymbol{g}}_k^{(1)}(\hat{\theta}_k), \tag{3.15a}$$

$$\hat{\theta}_k^{(I_{1,2})} \equiv \hat{\theta}_k^{(I_{1,1})} - a_k^{(1)}\, \hat{\boldsymbol{g}}_k^{(1)}\left(\hat{\theta}_k^{(I_{1,1})}\right), \tag{3.15b}$$

$$\hat{\theta}_k^{(I_{2,1})} \equiv \hat{\theta}_k^{(I_{1,2})} - a_k^{(2)}\, \hat{\boldsymbol{g}}_k^{(2)}\left(\hat{\theta}_k^{(I_{1,2})}\right), \tag{3.15c}$$

$$\hat{\theta}_k^{(I_{3,1})} \equiv \hat{\theta}_k^{(I_{2,1})} - a_k^{(1)}\, \hat{\boldsymbol{g}}_k^{(1)}\left(\hat{\theta}_k^{(I_{2,1})}\right), \tag{3.15d}$$

$$\hat{\theta}_k^{(I_{3,2})} \equiv \hat{\theta}_k^{(I_{3,1})} - a_k^{(1)}\, \hat{\boldsymbol{g}}_k^{(1)}\left(\hat{\theta}_k^{(I_{3,1})}\right), \tag{3.15e}$$

$$\hat{\theta}_{k+1} = \hat{\theta}_k^{(I_{3,2})}. \tag{3.15f}$$

Another way to visualize (3.15a–f) is given below:

$$\hat{\theta}_k \xrightarrow{(1)} \hat{\theta}_k^{(I_{1,1})} \xrightarrow{(1)} \hat{\theta}_k^{(I_{1,2})} \xrightarrow{(2)} \hat{\theta}_k^{(I_{2,1})} \xrightarrow{(1)} \hat{\theta}_k^{(I_{3,1})} \xrightarrow{(1)} \underbrace{\hat{\theta}_{k+1}}_{=\hat{\theta}_k^{(I_{3,2})}}, \tag{3.16}$$

where the numbers above the arrows indicate which subvector was updated. In (3.16) there are three main update "blocks": the first block consists of two updates to the first subvector (lines (3.15a,b)), the second block consists of a single update to the second subvector (line (3.15c)), and the third block consists of two more updates to the second subvector (lines (3.15d,e)). Next we provide a slight generalization of (3.15a–f) to allow for any general (deterministic) pattern for selecting the subvector to update. We begin by giving the following formal definition of what constitutes a "block" of subvector updates.

Definition 1. Let a block of subvector updates be defined as a *maximal consecutive sequence of updates on the same subvector*.

Let $s \geq 1$ denote the number of blocks in a single iteration (in the previous example we have $s = 3$) and let $n(m) \geq 1$ denote the number of updates in the mth block for $m = 1, \ldots, s$ (in the previous example, $n(1) = 2$, $n(2) = 1$, and $n(3) = 2$). Then, the vectors of the form $\hat{\theta}_k^{(I_{m,i})}$ in (3.15a–f) can be interpreted as the vector obtained during the mth block of the $(k+1)$st iteration after having performed exactly $i \geq 0$ updates within the block. Note that this interpretation implies $\hat{\theta}_k^{(I_{m,n(m)})} = \hat{\theta}_k^{(I_{m+1,0})}$ when $m < s$. Including this observation in (3.16):

$$\underbrace{\hat{\theta}_k}_{=\hat{\theta}_k^{(I_{1,0})}} \xrightarrow{(1)} \hat{\theta}_k^{(I_{1,1})} \xrightarrow{(1)} \underbrace{\hat{\theta}_k^{(I_{1,2})}}_{=\hat{\theta}_k^{(I_{2,0})}} \xrightarrow{(2)} \underbrace{\hat{\theta}_k^{(I_{2,1})}}_{=\hat{\theta}_k^{(I_{3,0})}} \xrightarrow{(1)} \hat{\theta}_k^{(I_{3,1})} \xrightarrow{(1)} \underbrace{\hat{\theta}_{k+1}}_{=\hat{\theta}_k^{(I_{3,2})}}.$$

While at first it may seem convoluted to assign two different labels to the same

vector, having $\hat{\theta}_k^{(I_{m,n(m)})} = \hat{\theta}_k^{(I_{m+1,0})}$ for $m < s$ serves to significantly simplify the presentation of the generalized cyclic SA algorithm of which (3.14) and (3.15a–f) are special cases.

An alternative way to write the algorithm in (3.15a–f) is as follows:

$$\begin{aligned} \boldsymbol{F}_k &\equiv \left[a_k^{(1)}\ \hat{\boldsymbol{g}}_k^{(1)}(\hat{\theta}_k) + a_k^{(2)}\ \hat{\boldsymbol{g}}_k^{(2)}\left(\hat{\theta}_k^{(I_{1,2})}\right) + \sum_{i=1}^{3} a_k^{(1)}\ \hat{\boldsymbol{g}}_k^{(1)}\left(\hat{\theta}_k^{(I_{i,1})}\right) \right] / a_k, \\ \hat{\theta}_{k+1} &= \hat{\theta}_k - a_k \boldsymbol{F}_k, \end{aligned} \tag{3.17}$$

for some sequence $\{a_i\}_{i\geq 0}$ with $a_i > 0$. Similarly, the algorithm in (3.14) can also be written in the form $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{F}_k$ after redefining the vector $\boldsymbol{F}_k$. Specifically, the algorithm in (3.14) can be written as follows:

$$\begin{aligned} &\text{Let } \mathcal{X}_k \sim \text{Bernoulli}(q_k), \\ &\text{Let } j = 1 \text{ if } \mathcal{X}_k = 1 \text{ and } j = 2 \text{ otherwise}, \\ &\boldsymbol{F}_k \equiv \left[\tilde{a}_k^{(j)} \hat{\boldsymbol{g}}_k^{(j)}(\hat{\theta}_k) \right] / a_k, \\ &\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{F}_k. \end{aligned} \tag{3.18}$$

In fact, (3.18) is a variant of (3.17) in which $s = 1$ (there is only one block per iteration), $n(1) = 1$ (there is only one update within the block), and the subvector to update within the block is selected at random.

In general, it is possible to conceive of a vector $\boldsymbol{F}_k$ with a more general form

than that in (3.17) or (3.18). The number of blocks in an iteration of (3.15a–f), for example, could be a random variable, s_k, depending on the iteration number. The number of updates within a block could also be a random variable, $n_k(m)$, depending on the iteration number as well as on the block number. Moreover, the choice of which subvector to update during the mth block could also be a random variable, $j_k(m)$, depending on k. These generalizations are captured by the generalized cyclic SA algorithm introduced next.

Let $\{\mathcal{S}_j\}_{j=1}^d$ be (not necessarily disjoint) subsets of $\mathcal{S} \equiv \{1, \ldots, p\}$ satisfying:

$$\bigcup_{j=1}^{d} \mathcal{S}_j = \mathcal{S}. \tag{3.19}$$

Next, for $j = 1, \ldots, d$ and any vector $\boldsymbol{v} = [v_1, \ldots, v_p]^\top \in \mathbb{R}^p$ define the vector $\boldsymbol{v}^{(j)}$, with ith entry denoted by $v_i^{(j)}$, as follows:

$$v_i^{(j)} = \begin{cases} v_i & \text{if } i \in \mathcal{S}_j, \\ 0 & \text{otherwise,} \end{cases} \tag{3.20}$$

this is simply a generalization of the notation used in (3.2) to the case where there are d subvectors possibly with shared components. Note that it is possible that $\sum_{j=1}^d \boldsymbol{g}^{(j)}(\theta) \neq \boldsymbol{g}(\theta)$ since $\{\mathcal{S}_j\}_{j=1}^d$ are not necessarily disjoint. The generalized cyclic stochastic approximation (GCSA) algorithm is described next.

Using the notation in (3.20), the idea behind GCSA is as follows: given the

Algorithm 1 Generalized Cyclic Stochastic Approximation (GCSA)

Require: $\hat{\theta}_0$ and $\{a_k^{(j)}\}_{k\geq 0}$ for $j = 1, \ldots, d$. Set $k = 0$.

1: **while** stopping criterion has not been reached **do**

2: Let $s_k \in \mathbb{Z}^+$ be a random variable, where $\mathbb{Z}^+$ denotes the positive integers.

3: **for** $m = 1, \ldots, s_k$ **do**

4: Let $j_k(m) \in \{1, \ldots, d\}$ be a random variable, where d is defined above (3.19) (recall d is the number of subvectors that can be updated).

5: Let $n_k(m) \in \mathbb{Z}^+$ be a random variable.

6: **for** $i = 1, \ldots, n_k(m)$ **do**

7: Define:

$$\hat{\theta}_k^{(I_{m,i})} \equiv \hat{\theta}_k - \sum_{z=1}^{m-1} \sum_{\ell=0}^{n_k(z)-1} \left[\tilde{A}_k(z)\, \hat{\boldsymbol{g}}_k^{(j_k(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) \right] - \sum_{\ell=0}^{i-1} \left[\tilde{A}_k(m)\, \hat{\boldsymbol{g}}_k^{(j_k(m))} \left(\hat{\theta}_k^{(I_{m,\ell})} \right) \right],$$

where $\tilde{A}_k(m) \equiv \sum_{j=1}^{d} \chi\{j_k(m) = j\} \tilde{a}_k^{(j)}$, $\tilde{a}_k^{(j)}$ is the first unused element of the predetermined sequence $\{a_k^{(j)}\}_{k\geq 0}$, $\chi\{\mathcal{E}\}$ denotes the indicator function of the event $\mathcal{E}$, and $\sum_{i=a}^{b}(\cdot)_i = 0$ whenever $b < a$. A mathematical definition of the random sequence $\tilde{a}_k^{(j)}$ is given in (3.22) and Figure 3.2 gives a visual representation of the process through which $\tilde{a}_k^{(j)}$ is updated.

8: **end for**

9: **end for**

10: Let:

$$\hat{\theta}_{k+1} = \hat{\theta}_k^{(I_{m,i})} \text{ with } m = s_k \text{ and } i = n_k(s_k).$$

11: set $k = k + 1$

12: **end while**

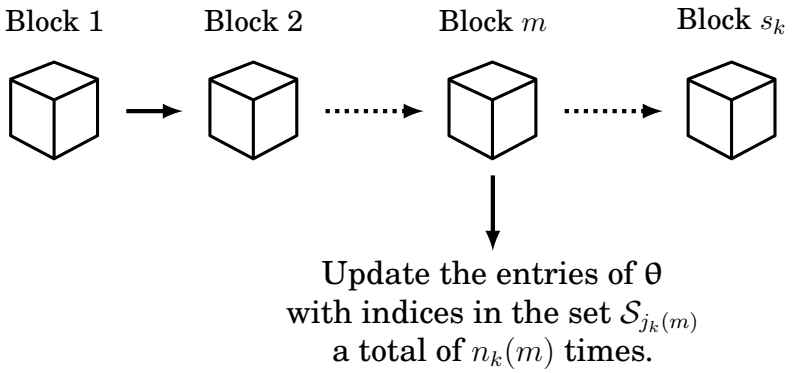


Figure 3.1: The components of the $(k+1)$st iteration of the GCSA algorithm (Algorithm 1). The numbers $j_k(m)$, $n_k(m)$, and s_k are allowed to be random variables. Once the value of $j_k(m)$ has been obtained, the entries to update are determined by the indices in $\mathcal{S}_{j_k(m)}$. The updates performed within the mth block correspond to lines 6–8 of Algorithm 1.

current parameter estimate, $\hat{\theta}_k$, a decision is first made on which of its coordinates to update by selecting the coordinates in one of the sets $\mathcal{S}_1, \ldots, \mathcal{S}_d$ according to a random variable. The number of blocks, number of updates within each block, and the subvector to update within each block are also random variables. After $\hat{\theta}_{k+1}$ has been obtained, the process is repeated until a stopping criterion is reached. The process needed to update $\hat{\theta}_k$ to $\hat{\theta}_{k+1}$ will constitute an *iteration* of GCSA. Figure 3.1 gives a representation of the components of an iteration of GCSA and Algorithm 1 provides an outline of GCSA.

Let us define the sequence $\tilde{a}_k^{(j)}$ in Algorithm 1 more precisely. First, define:

$$\varphi_k^{(j)} \equiv \sum_{i=0}^{k} \chi\left\{\left(\sum_{m=1}^{s_i} \chi\{j_i(m) = j\}\right) > 0\right\} - 1, \tag{3.21}$$

for each j. In other words, to compute $\varphi_k^{(j)}$ one must subtract one from the

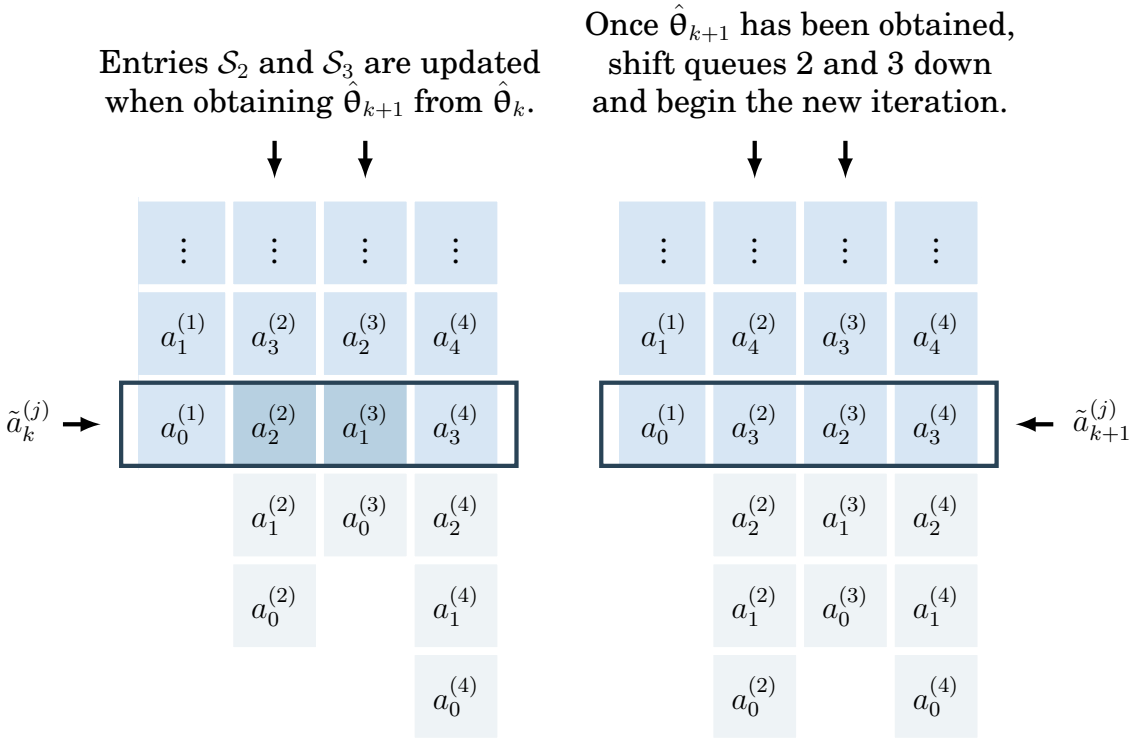


Figure 3.2: This is an illustration of the process through which $\tilde{a}_k^{(j)}$ is updated. For $j = 1, \ldots, 4$, the jth column of each of the two "tables" above represents a queue containing the used and unused elements of $\{a_i^{(j)}\}_{i \geq 0}$. In the example above, initially $\tilde{a}_k^{(1)} = a_0^{(1)}$, $\tilde{a}_k^{(2)} = a_2^{(2)}$, $\tilde{a}_k^{(3)} = a_1^{(3)}$, $\tilde{a}_k^{(4)} = a_3^{(4)}$. Then, the subvectors with entries in the sets $\mathcal{S}_2$ and $\mathcal{S}_3$ are updated when obtaining $\hat{\theta}_{k+1}$ from $\hat{\theta}_k$. Consequently, $\tilde{a}_{k+1}^{(1)} = a_0^{(1)}$, $\tilde{a}_{k+1}^{(2)} = a_3^{(2)}$, $\tilde{a}_{k+1}^{(3)} = a_2^{(3)}$, and $\tilde{a}_{k+1}^{(4)} = a_3^{(4)}$.

number of iterations up to and including iteration $k+1$ that use the sequence $\{a_i^{(j)}\}_{i \geq 0}$. Then, for $k+1 \geq 0$ the value of $\tilde{a}_{k+1}^{(j)}$ satisfies the following equation:

$$\tilde{a}_{k+1}^{(j)} = a_0^{(j)} + \sum_{i=0}^{k} \chi\{\varphi_k^{(j)} \geq i\}(a_{i+1}^{(j)} - a_i^{(j)}). \tag{3.22}$$

Note that $\tilde{a}_0^{(j)} = a_0^{(j)}$ for all j since $\sum_a^b(\cdot) = 0$ if $a > b$. Note also that if the sequence $\{a_i^{(j)}\}_{i \geq 0}$ is not used during the $(k+1)$st iteration, that is if $j_k(m) \neq j$

for all m, then $\tilde{a}_{k+1}^{(j)}$ will be equal to $\tilde{a}_k^{(j)}$. This shows that $\tilde{a}_{k+1}^{(j)}$ and the indicator function $\chi\{j_k(m) \neq j\}$ may be *dependent random variables* (with the exception of the case where $\tilde{a}_k^{(j)}$ is a deterministic function of k, as in the case of (3.17)).

The expression for $\hat{\theta}_{k+1} = \hat{\theta}_k^{(I_{s_k, n_k(s_k)})}$ in the GCSA algorithm (Algorithm 1) is equivalent to the recursion:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{F}_k, \text{ where } \boldsymbol{F}_k \equiv \left[\sum_{z=1}^{s_k} \sum_{\ell=0}^{n_k(z)-1} \tilde{A}_k(z)\, \hat{\boldsymbol{g}}_k^{(j_k(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) \right] / a_k, \quad (3.23)$$

and where $a_k > 0$. Therefore, an iteration of the GCSA algorithm can be written in the general form $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{F}_k$ (the two algorithms defined by (3.17) and (3.18) are special cases of (3.23)). Although (3.23) resembles the general SA update in (2.3), existing results on convergence of SA algorithms are not directly applicable to GCSA due to the increased complexity of $\boldsymbol{F}_k$ over $\boldsymbol{Y}_k$, the typical SA update direction.

3.5 Concluding Remarks

This chapter described the GCSA algorithm for stochastic optimization via SA, the iterative algorithm for updating the parameter vector $\hat{\theta}_k$ that will be the focus of this dissertation. The basic idea behind GCSA is to divide the vector of parameters into d (possibly overlapping) subvectors. Then, at each time a decision is made regarding which subvector to update (this decision can be

made according to a random variable or may be governed by a deterministic selection pattern). The subvector updates are performed using SA-based update directions (in general, it is impossible to guarantee that the update direction is a descent direction due to the presence of the noise and bias terms described in Section 2.2). In the GCSA algorithm, each subvector has an associated gain sequence that is a deterministic function of k and is used to scale the update direction. Although the gain sequences for the different subvectors may be different, the convergence theory in the following chapter will require that all gain sequences converge to zero at the same rate.

Chapter 4

Convergence of GCSA

This chapter derives conditions for the convergence w.p.1 of the GCSA iterates. Section 4.1 first rewrites the GCSA algorithm as a stochastic time-dependent process. Section 4.2 then provides a detailed analysis of the process from Section 4.1. Section 4.3 states the main theorem for this chapter (Theorem 2). Section 4.4 states two corollaries regarding special cases of GCSA. Section 4.5 discusses the validity of the assumptions of Theorem 2. Lastly, Section 4.6 contains concluding remarks.

4.1 Rewriting the GCSA Recursion

The theory behind the convergence of GCSA relies on rewriting the algorithm as a stochastic time-dependent process. Loosely speaking, this section first rewrites a realization of GCSA as a multi-dimensional, continuous, time-

dependent function. This continuous time-dependent function is then shown to satisfy a time-dependent version of the GCSA recursion in (3.23), a fact which will play a crucial role in the development of the convergence theory for GCSA in Sections 4.2 and 4.3. The aforementioned time-dependent *continuous function* and a related time-dependent *step function* are constructed next.

We begin by defining the following time-dependent *step function*:

$$\bar{\boldsymbol{Z}}_0(t) \equiv \begin{cases} \hat{\theta}_k & \text{if } t \in [t_k, t_{k+1}) \text{ for } k \geq 0, \\ \bar{\boldsymbol{Z}}_0(t_0) & \text{if } t \leq t_0, \end{cases} \tag{4.1}$$

where $t_k \equiv \sum_{i=0}^{k-1} a_i$ for $k \geq 0$ and where $a_k > 0$ is a deterministic sequence satisfying $a_k \to 0$ (Section 4.2 imposes other assumptions on a_k relating it to the gain sequences $a_k^{(j)}$ from Algorithm 1). Now, the time-dependent *continuous function* discussed in this section's opening paragraph will be defined as:

$$\boldsymbol{Z}_0(t) \equiv \frac{(t_{k+1} - t)}{a_k} \bar{\boldsymbol{Z}}_0(t_k) + \frac{(t - t_k)}{a_k} \bar{\boldsymbol{Z}}_0(t_{k+1}) \tag{4.2}$$

for $t \in [t_k, t_{k+1}]$, and $\boldsymbol{Z}_0(t) \equiv \boldsymbol{Z}_0(t_0)$ for $t \leq t_0$. The function $\boldsymbol{Z}_0(t)$ is then simply an interpolation of $\bar{\boldsymbol{Z}}_0(t)$ at the interpolation points $\{t_k\}_{k \geq 0}$ (see Figure 4.1). The subindex "0" is used in anticipation of modified versions of $\bar{\boldsymbol{Z}}_0(t)$ and $\boldsymbol{Z}_0(t)$ to be introduced in Section 4.2. Next we rewrite the GCSA recursion in (3.23) by decomposing $\boldsymbol{F}_k$ into several terms, after which the resulting expression is

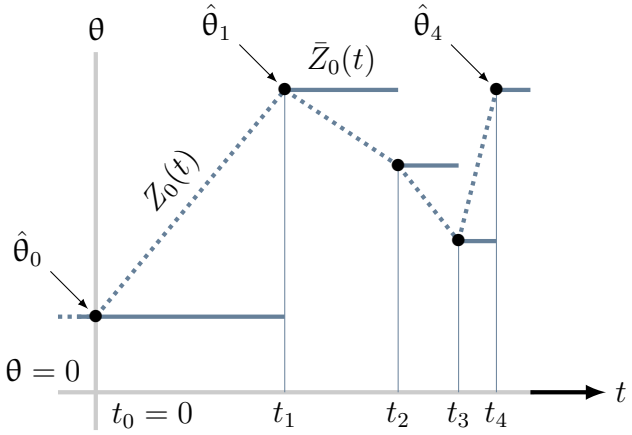


Figure 4.1: The difference between $\bar{Z}_0(t)$ and $Z_0(t)$, special cases of $\bar{\boldsymbol{Z}}_0(t)$ and $\boldsymbol{Z}_0(t)$ where both functions are real-valued. In general, $\bar{\boldsymbol{Z}}_0(t_k) = \boldsymbol{Z}_0(t_k) = \hat{\theta}_k$.

used to construct a time-dependent analogue to (3.23) involving $\boldsymbol{Z}_0(t)$.

By adding and subtracting select terms (effectively adding zero) to the definition of $\boldsymbol{F}_k$ in (3.23) and using the fact that $\hat{\boldsymbol{g}}_k(\theta) = \boldsymbol{g}(\theta) + \beta_k(\theta) + \xi_k(\theta)$ (see Section 2.2), the recursion in (3.23) can first be rewritten as:

$$\begin{aligned}\hat{\theta}_{k+1} = \hat{\theta}_k &- \sum_{m=1}^{s_k} \sum_{i=0}^{n_k(m)-1} \tilde{A}_k(m) \left[\beta_k^{(j_k(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right) + \xi_k^{(j_k(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right)\right] \\ &- \sum_{m=1}^{s_k} \sum_{i=0}^{n_k(m)-1} \tilde{A}_k(m) \left[\boldsymbol{g}^{(j_k(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right) - \boldsymbol{g}^{(j_k(m))}(\hat{\theta}_k)\right] \\ &- \sum_{m=1}^{s_k} \sum_{i=0}^{n_k(m)-1} \tilde{A}_k(m)\, \boldsymbol{g}^{(j_k(m))}(\hat{\theta}_k). \end{aligned} \tag{4.3}$$

Next, for $j = 1, \ldots, d$ define the random variables:

$$x_k(j) \equiv \left(\frac{\tilde{a}_k^{(j)}}{a_k}\right) \sum_{m=1}^{s_k} \chi\{j_k(m) = j\} n_k(m). \tag{4.4}$$

Assuming $\mu_k(j) \equiv E[x_k(j)]$ exists and is finite and that $\mu(j) \equiv \lim_{k\to\infty} \mu_k(j)$ exists and is finite (these assumptions will be discussed in Section 4.5), define:

$$\boldsymbol{h}_k(\theta) \equiv \sum_{j=1}^{d} \mu_k(j)\boldsymbol{g}^{(j)}(\theta), \quad \boldsymbol{h}(\theta) \equiv \sum_{j=1}^{d} \mu(j)\boldsymbol{g}^{(j)}(\theta). \tag{4.5}$$

By adding and subtracting $a_k \boldsymbol{h}(\hat{\theta}_k)$ from the right-hand side of (4.3):

$$\begin{aligned}\hat{\theta}_{k+1} = \hat{\theta}_k - a_k\boldsymbol{h}(\hat{\theta}_k) - a_k\left[\sum_{m=1}^{s_k}\sum_{i=0}^{n_k(m)-1}\left(\frac{\tilde{A}_k(m)}{a_k}\right)\boldsymbol{g}^{(j_k(m))}(\hat{\theta}_k) - \boldsymbol{h}(\hat{\theta}_k)\right] \\ - a_k\sum_{m=1}^{s_k}\sum_{i=0}^{n_k(m)-1}\left(\frac{\tilde{A}_k(m)}{a_k}\right)\left[\beta_k^{(j_k(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right) + \xi_k^{(j_k(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right)\right] \\ - a_k\sum_{m=1}^{s_k}\sum_{i=0}^{n_k(m)-1}\left(\frac{\tilde{A}_k(m)}{a_k}\right)\left[\boldsymbol{g}^{(j_k(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right) - \boldsymbol{g}^{(j_k(m))}(\hat{\theta}_k)\right]. \end{aligned} \tag{4.6}$$

Thus, when $\mu_k(j)$ and $\mu(j)$ exist and are finite, (4.6) is equivalent to the GCSA recursion from (3.23). Next we derive a time-dependent version of (4.6).

We begin by defining the following terms:

$$\bar{\boldsymbol{B}}_0(t_k) \equiv \sum_{r=0}^{k-1} a_r \sum_{m=1}^{s_r}\sum_{i=0}^{n_r(m)-1}\left(\frac{\tilde{A}_r(m)}{a_r}\right)\beta_r^{(j_r(m))}\left(\hat{\theta}_r^{(I_{m,i})}\right), \tag{4.7a}$$

$$\bar{\boldsymbol{M}}_0(t_k) \equiv \sum_{r=0}^{k-1} a_r \sum_{m=1}^{s_r}\sum_{i=0}^{n_r(m)-1}\left(\frac{\tilde{A}_r(m)}{a_r}\right)\xi_r^{(j_r(m))}\left(\hat{\theta}_r^{(I_{m,i})}\right), \tag{4.7b}$$

$$\bar{\boldsymbol{N}}_0(t_k) \equiv \sum_{r=0}^{k-1} a_r \left[\sum_{m=1}^{s_r}\sum_{i=0}^{n_r(m)-1}\left(\frac{\tilde{A}_r(m)}{a_r}\right)\boldsymbol{g}^{(j_r(m))}(\hat{\theta}_r) - h(\hat{\theta}_r)\right], \tag{4.7c}$$

$$\bar{\boldsymbol{W}}_0(t_k) \equiv \sum_{r=0}^{k-1} a_r \sum_{m=1}^{s_r}\sum_{i=0}^{n_r(m)-1}\left(\frac{\tilde{A}_r(m)}{a_r}\right)\left[\boldsymbol{g}^{(j_r(m))}\left(\hat{\theta}_r^{(I_{m,i})}\right) - \boldsymbol{g}^{(j_r(m))}(\hat{\theta}_r)\right]. \tag{4.7d}$$

Then, using the notation from (4.7a–d), an equivalent way to write (4.6) is:

$$\begin{aligned}\bar{\boldsymbol{Z}}_0(t_{k+1}) = \bar{\boldsymbol{Z}}_0(t_0) - \bar{\boldsymbol{B}}_0(t_{k+1}) - \bar{\boldsymbol{M}}_0(t_{k+1}) \\ - \bar{\boldsymbol{N}}_0(t_{k+1}) - \bar{\boldsymbol{W}}_0(t_{k+1}) - \sum_{i=0}^{k} a_i \boldsymbol{h}(\bar{\boldsymbol{Z}}_0(t_i)).\end{aligned} \tag{4.8}$$

The expression in (4.8) lays the foundation for constructing the continuous and time-dependent version of the GCSA recursion (4.6). Specifically, in a manner analogous to (4.1) let $\bar{\boldsymbol{B}}_0(t)$, $\bar{\boldsymbol{M}}_0(t)$, $\bar{\boldsymbol{N}}_0(t)$, and $\bar{\boldsymbol{W}}_0(t)$ be step functions on the interval $[t_k, t_{k+1})$ and, in a manner analogous to (4.2), let $\boldsymbol{B}_0(t)$, $\boldsymbol{M}_0(t)$, $\boldsymbol{N}_0(t)$, and $\boldsymbol{W}_0(t)$ denote their respective interpolation functions. Then, using $t_0 = 0$:

$$\boldsymbol{Z}_0(t) = \boldsymbol{Z}_0(0) - \boldsymbol{B}_0(t) - \boldsymbol{M}_0(t) - \boldsymbol{N}_0(t) - \boldsymbol{W}_0(t) - \int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_0(s))\, ds. \tag{4.9}$$

Alternatively, replacing the integrand (4.9) with $\boldsymbol{h}(\boldsymbol{Z}_0(s))$ we obtain:

$$\boldsymbol{Z}_0(t) = \boldsymbol{Z}_0(0) - \boldsymbol{B}_0(t) - \boldsymbol{M}_0(t) - \boldsymbol{N}_0(t) - \boldsymbol{W}_0(t) - \int_0^t \boldsymbol{h}(\boldsymbol{Z}_0(s))\, ds + \boldsymbol{\zeta}_0(t), \tag{4.10}$$

where $\boldsymbol{\zeta}_0(t)$ is a vector representing the error introduced by using $\boldsymbol{h}(\boldsymbol{Z}_0(s))$ in place of $\boldsymbol{h}(\bar{\boldsymbol{Z}}_0(s))$. Equation (4.10) represents a continuous and time-dependent version of the GCSA recursion. The following section proves a few lemmas regarding the GCSA algorithm when treated as the stochastic time-dependent, continuous process in (4.10).

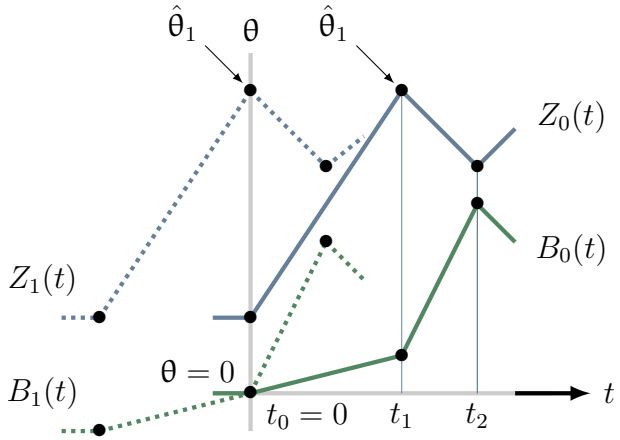


Figure 4.2: Comparing $\boldsymbol{Z}_0(t)$ to $\boldsymbol{Z}_1(t)$ and $\boldsymbol{B}_0(t)$ to $\boldsymbol{B}_1(t)$ for the case where all functions have real-valued output. The variables in (4.12a,b) are obtained by shifting a function up/down and to the left so that $\boldsymbol{B}_k(t_0) = \boldsymbol{M}_k(t_0) = \boldsymbol{N}_k(t_0) = \boldsymbol{W}_k(t_0) = 0$. In contrast, the functions in (4.11) are obtained solely via left-shifts so that $\bar{\boldsymbol{Z}}_k(t_0) = \boldsymbol{Z}_k(t_0) = \hat{\theta}_k$.

4.2 Analyzing the GCSA Recursion

This section proves several results regarding a set of shifted versions of (4.9) and (4.10). Specifically, define the *shift functions*:

$$\bar{\boldsymbol{Z}}_k(t) \equiv \bar{\boldsymbol{Z}}_0(t_k + t), \quad \boldsymbol{Z}_k(t) \equiv \boldsymbol{Z}_0(t_k + t), \tag{4.11}$$

and the *shift-increment functions*:

$$\boldsymbol{B}_k(t) \equiv \boldsymbol{B}_0(t_k + t) - \boldsymbol{B}_0(t_k), \quad \boldsymbol{M}_k(t) \equiv \boldsymbol{M}_0(t_k + t) - \boldsymbol{M}_0(t_k), \tag{4.12a}$$

$$\boldsymbol{N}_k(t) \equiv \boldsymbol{N}_0(t_k + t) - \boldsymbol{N}_0(t_k), \quad \boldsymbol{W}_k(t) \equiv \boldsymbol{W}_0(t_k + t) - \boldsymbol{W}_0(t_k). \tag{4.12b}$$

(Figure 4.2 illustrates the difference between a shift function and a shift-

increment function.) Using the notation from (4.11) and (4.12a,b), $\boldsymbol{Z}_k(t)$ satisfies:

$$\boldsymbol{Z}_k(t) = \boldsymbol{Z}_k(0) - \boldsymbol{B}_k(t) - \boldsymbol{M}_k(t) - \boldsymbol{N}_k(t) - \boldsymbol{W}_k(t) - \int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\, ds. \tag{4.13}$$

Alternatively, replacing the integrand (4.13) with $\boldsymbol{h}(\boldsymbol{Z}_k(s))$:

$$\boldsymbol{Z}_k(t) = \boldsymbol{Z}_k(0) - \boldsymbol{B}_k(t) - \boldsymbol{M}_k(t) - \boldsymbol{N}_k(t) - \boldsymbol{W}_k(t) - \int_0^t \boldsymbol{h}(\boldsymbol{Z}_k(s))\, ds + \boldsymbol{\zeta}_k(t), \tag{4.14}$$

for a vector $\boldsymbol{\zeta}_k(t)$ representing the error introduced by using $\boldsymbol{h}(\boldsymbol{Z}_k(s))$ in place of $\boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))$. Equation (4.14) is similar to equation (2.3.3) in Kushner and Clark (1978), though (4.14) is significantly more complex. This section proves several lemmas pertaining to terms appearing in (4.13) and (4.14). These lemmas will then be used in Section 4.3 to prove this chapter's main convergence theorem. Pages 265– 268 contain a compilation of frequently used notation (including a section with GCSA-specific notation) which may be used as a quick reference.

The results in this section make use of Kolmogorov's Extension Theorem (Øksendal 2003, Theorem 2.1.5), which guarantees that any realization of GCSA can be seen as a random variable on a probability space $(\Omega, \mathcal{F}, P)$, where Ω is the sample space, $\mathcal{F}$ is the σ-field and P is a probability measure. In other words, each $\omega \in \Omega$ is assumed to determine (via some unknown function)

an entire realization of GCSA. Thus, for example, the random sets $\{\tilde{A}_k^{(j_k(m))}\}_k$, $\{\hat{\theta}_k^{(I_{m,i})}\}_{k,m,i}$, $\{\xi_k^{(j_k(m))}(\hat{\theta}_k^{(I_{m,i})})\}_{k,m,i}$, and $\{\beta_k^{(j_k(m))}(\hat{\theta}_k^{(I_{m,i})})\}_{k,m,i}$ are fully determined by ω. Next, we introduce a set of assumptions to be used throughout this section. Throughout these assumptions let ω be as defined above.

A0 Let $a_k > 0$ and $a_k^{(j)} > 0$ for all k and j. Let $\sum_{k=0}^{\infty} a_k = \infty$ and $\sum_{k=0}^{\infty} a_k^2 < \infty$. Next, let $E[\tilde{a}_k^{(j)}/a_k] \to r_j$ with $0 \leq r_j < \infty$. Additionally, let there exist a set $\Omega_0 \subset \Omega$ with $P(\Omega_0) = 1$ such that $\tilde{a}_k^{(j)}/a_k \to r_j$ for all $\omega \in \Omega_0$.

A1 Let s_k and $n_k(m)$ be bounded uniformly over k, m, and ω.

A2 Assume $\mu_k(j) = E[x_k(j)]$ exists for all k and j and is finite (i.e., $\mu_k(j) < \infty$) and that $\mu(j) = \lim_{k\to\infty} \mu_k(j)$ exists and is finite ($x_k(j)$, $\mu_k(j)$, and $\mu(j)$ were first introduced in pp. 52–53). Additionally, define:

$$C_k(j) \equiv \sum_{m=1}^{s_k} \chi\{j_k(m) = j\} n_k(m), \quad S_k \equiv \sum_{j=1}^{d} E\left[\frac{\tilde{a}_k^{(j)}}{a_k}\right] (C_k(j) - E[C_k(j)]),$$

and let $E[S_k S_\ell] = 0$ for $k \neq \ell$. Furthermore, let $\tilde{a}_k^{(j)}$ and $C_k(j)$ be independent.

A3 Let $g(\theta)$ be a continuous function (i.e., $L(\theta)$ is continuously differentiable).

A4 For $k \geq 0$, $m \leq s_k$, $i \leq n_k(m)$, let there exist a set $\Omega_1 \subset \Omega$ with $P(\Omega_1) = 1$ and a scalar $0 < R_1(\omega) < \infty$ such that the set $\{\hat{\theta}_k^{(I_{m,i})}\}_{k,m,i}$ is contained within a p-dimensional ball of radius $R_1(\omega)$ centered at the origin for all $\omega \in \Omega_1$.

A5 For $k \geq 0$, $m \leq s_k$, $i \leq n_k(m)$, let there exist a set $\Omega_2 \subset \Omega$ with $P(\Omega_2) = 1$ and a scalar $0 < R_2(\omega) < \infty$ such that the set $\{\beta_k^{(j_k(m))}(\hat{\theta}_k^{(I_{m,i})})\}_{k,m,i}$ is contained within a p-dimensional ball of radius $R_2(\omega)$ centered at the origin for all $\omega \in \Omega_2$. Additionally, let $\beta_k^{(j_k(m))}(\hat{\theta}_k^{(I_{m,i})})$ converge to **0** w.p.1 as $k \to \infty$.

A6 Define $\boldsymbol{D}_r \equiv \sum_{m=1}^{s_r} \sum_{i=0}^{n_r(m)-1} \left(\frac{\tilde{A}_r(m)}{a_r}\right) \xi_r^{(j_r(m))} \left(\hat{\theta}_r^{(I_{m,i})}\right)$. Assume for all $\varepsilon > 0$ we have $\lim_{k\to\infty} P\left(\sup_{m\geq k} \left\| \sum_{r=k}^{m} a_r \boldsymbol{D}_r \right\| \geq \varepsilon\right) = 0$.

A7 Let θ^* be a locally asymptotically stable (in the sense of Lyapunov) solution of the differential equation:

$$\dot{\boldsymbol{Z}}(t) \equiv \frac{d\boldsymbol{Z}(t)}{dt} = -\sum_{j=1}^{d} \mu(j) \boldsymbol{g}^{(j)}(\boldsymbol{Z}(t)) \tag{4.15}$$

with domain of attraction $DA(\theta^*)$.

A8 There is a compact subset A of $DA(\theta^*)$ such that $\hat{\theta}_k \in A$ infinitely often (here compactness is defined using the Euclidean topology).

The validity of A0–A8 will be discussed in Sections 4.4 and 4.5. We note that while a_k is not used by the GCSA algorithm, it serves as a representation of the rate at which $a_k^{(j)}$ decreases. Following are several lemmas regarding the set of functions $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$. Throughout all proofs any unspecified probabilistic arguments are meant to hold w.p.1.

Lemma 1. Assume A0 and A1 hold. Using the definition of $x_k(j)$ in (4.4) let:

$$X_k \equiv \sum_{j=1}^{d} x_k(j). \tag{4.16}$$

Then, there exists a constant $0 < R_3(\omega) < \infty$ such that $|X_k| < R_3(\omega)$ for all $\omega \in \Omega_3$ and all $k \geq 0$.

Proof. By A1 it follows that s_k and $n_k(m)$ are bounded (uniformly in k, m, and ω). Next, A0 (using the fact that $\tilde{a}_k^{(j)}/a_k \to r_j$ w.p.1 where $0 \leq r_j < \infty$) implies that for $\omega \in \Omega_0$ the sequence $\{\tilde{a}_k^{(j)}/a_k\}_{k\geq 0}$ is bounded in magnitude by a constant that depends on ω but is independent of k. Combining this with the definition of $x_k(j)$ given in (4.4) and the fact that $d < \infty$ yields the desired result. □

Lemma 2. Assume condition A4 holds. Then, w.p.1 the set $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ is a set of continuous functions that are bounded over t uniformly in k.

Proof. A direct consequence of condition A4 is that $\{\hat{\theta}_k\}_{k\geq 0}$ must be a bounded set (although the magnitude of the bound may be a function of ω). Since $\boldsymbol{Z}_0(t)$ was obtained by interpolating the vectors $\{\hat{\theta}_k\}_{k\geq 0}$ then $\boldsymbol{Z}_0(t)$ must be bounded uniformly over $t \in \mathbb{R}$ w.p.1. Since $\boldsymbol{Z}_k(t)$ is obtained by performing a sequence of left-shifts on $\boldsymbol{Z}_0(t)$, all functions in $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ must be bounded uniformly in k and t w.p.1. Additionally, these functions are clearly continuous by construction (the functions are linear interpolations of bounded vectors). □

The next lemma relies on the concept of the "equicontinuity" of set of functions. This concept and two related concepts are defined next.

Definition 2. A set $\{\boldsymbol{\rho}_k(t)\}_{k\geq 0}$ of functions from $\mathbb{R}$ to $\mathbb{R}^p$ is said to be *equicontinuous* at t if for all k and $\epsilon > 0$ there exists a $\delta(t,\epsilon)$ such that $\|\boldsymbol{\rho}_k(t) - \boldsymbol{\rho}_k(s)\| < \epsilon$ if $|t-s| < \delta(t,\epsilon)$. If the set of functions is equicontinuous at every t it is said to be *point-wise equicontinuous*. The set of functions is said to be *uniformly equicontinuous* over a set $\mathcal{S} \subset \mathbb{R}$ if for all $k \geq 0$, $t \in \mathcal{S}$, and $\epsilon > 0$ there exists a $\delta(\epsilon)$ such that $\|\boldsymbol{\rho}_k(t) - \boldsymbol{\rho}_k(s)\| < \epsilon$ whenever $|t-s| < \delta(\epsilon)$ and $s \in \mathcal{S}$ (note that $\delta(\epsilon)$ depends neither on k nor on t).

Lemma 3. Assume conditions A0, A1, and A5 hold and for any finite $T > 0$ define $I_T \equiv [-T, T]$. Then, the following statements hold for all ω in a set w.p.1 and any finite $T > 0$. The set $\{\boldsymbol{B}_k(t)\}_{k\geq 0}$ is a set of functions, each of which is uniformly continuous over I_T (note that at this point we are not claiming equicontinuity). Additionally, the functions in $\{\boldsymbol{B}_k(t)\}_{k\geq 0}$ are bounded over $t \in I_T$ uniformly in k. Furthermore, $\boldsymbol{B}_k(t) \to \mathbf{0}$ uniformly over $t \in I_T$ as $k \to \infty$. The former statements imply $\{\boldsymbol{B}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous over I_T.

Proof. First, we prove that $\{\boldsymbol{B}_k(t)\}_{k\geq 0}$ is bounded over $t \in I_T$ uniformly in k. First and foremost, using the fact that all gain sequences are nonnegative (condition A0), the definition of $\tilde{A}_r(m)$ given in Algorithm 1, and the definition of $\bar{\boldsymbol{B}}_0(t_k)$ in (4.7a) along with the triangle inequality gives rise to the following

inequality:

$$\|\bar{\boldsymbol{B}}_0(t_k)\| \leq \sum_{r=0}^{k-1} a_r \sum_{m=1}^{s_r} \sum_{i=0}^{n_r(m)-1} \left(\frac{\tilde{A}_r(m)}{a_r} \right) \left\| \boldsymbol{\beta}_r^{(j_r(m))} \left(\hat{\theta}_r^{(I_{m,i})} \right) \right\|. \tag{4.17}$$

Under A5, for all $\omega \in \Omega_2$ there exists a constant $R_2(\omega)$ such that the term $\|\boldsymbol{\beta}_k^{(j_k(m))} \left(\hat{\theta}_k^{(I_{m,i})} \right)\|$ is bounded above by $R_2(\omega)$. Then, from (4.17) we have:

$$\begin{aligned} \|\bar{\boldsymbol{B}}_0(t_{k+1}) - \bar{\boldsymbol{B}}_0(t_k)\| &\leq a_k \sum_{m=1}^{s_k} \sum_{i=0}^{n_k(m)-1} \left(\frac{\tilde{A}_k(m)}{a_k} \right) R_2(\omega) \\ &= a_k X_k R_2(\omega) \end{aligned}$$

for all $\omega \in \Omega_2$, where X_k was defined in (4.16). Using the result from Lemma 1 we obtain the bound $|X_k| < R_3(\omega)$ for $\omega \in \Omega_3$, which implies:

$$\|\bar{\boldsymbol{B}}_0(t_{k+1}) - \bar{\boldsymbol{B}}_0(t_k)\| \leq a_k R_2(\omega) R_3(\omega) \tag{4.18}$$

for all $\omega \in \Omega_2 \cap \Omega_3$ (note that $P(\Omega_2 \cap \Omega_3) = 1$). Because $\boldsymbol{B}_0(t)$ is a piecewise-linear interpolation of $\bar{\boldsymbol{B}}_0(t)$ at the points $\{t_k\}_{k \geq 0}$, it follows from (4.18) that for $t \in \mathbb{R}$:

$$\|\boldsymbol{B}_k(t)\| = \|\boldsymbol{B}_0(t_k + t) - \boldsymbol{B}_0(t_k)\| \leq |t| R_2(\omega) R_3(\omega). \tag{4.19}$$

The inequality in (4.19) implies that each function in $\{\boldsymbol{B}_k(t)\}_{k \geq 0}$ is bounded in magnitude by $T R_2(\omega) R_3(\omega)$ for $t \in I_T$ and $\omega \in \Omega_2 \cap \Omega_3$ (independently of k).

Observe now that $\boldsymbol{B}_k(t)$ is continuous by construction. Since any continuous function on a compact set is uniformly continuous on that set, for each k the function $\boldsymbol{B}_k(t)$ is uniformly continuous on I_T (Rudin 1976, Theorem 4.19). So far, we have proven that the functions in the set $\{\boldsymbol{B}_k(t)\}_{k\geq 0}$ are bounded uniformly (in k) for $t \in I_T$ and that, for each k, the function $\boldsymbol{B}_k(t)$ is uniformly continuous over $t \in I_T$. Next, we show $\boldsymbol{B}_k(t)$ converges to the zero vector uniformly over $t \in I_T$.

By the last part of A5 (the convergence of the bias vector to zero), there exists a set $\Omega_4 \subset \Omega$ with $P(\Omega_4) = 1$ such that for any $\omega \in \Omega_4$ and any $\epsilon > 0$ there exists a finite constant $K_1(\omega, \epsilon)$ such that $\|\boldsymbol{\beta}_k^{(j_k(m))}(\hat{\boldsymbol{\theta}}_k^{(I_{m,i})})\| \leq \epsilon$ whenever $k \geq K_1(\omega, \epsilon)$. Therefore, using (4.19) with $R_2(\omega)$ replaced by ϵ yields the bound:

$$\|\boldsymbol{B}_k(t)\| = \|\boldsymbol{B}_0(t_k + t) - \boldsymbol{B}_0(t_k)\| \leq \epsilon|t|R_3(\omega), \tag{4.20}$$

provided $t_k + t > t_{K_1(\omega,\epsilon)}$. Note that having $k > K_1(\omega, \epsilon)$ does not guarantee that $t_k + t > t_{K_1(\omega,\epsilon)}$ since t may be negative. However, by condition A0 (using the fact that $\sum_{k=0}^{\infty} a_k = \infty$) there exists a finite constant $K_2(T, K_1(\omega, \epsilon))$ with $K_2(T, K_1(\omega, \epsilon)) \geq K_1(\omega, \epsilon)$ such that for $k \geq K_2(T, K_1(\omega, \epsilon))$ and $t \in I_T$ we have $t_k + t \geq t_k - T \geq t_{K_1(\omega,\epsilon)}$. In other words, if k is large enough (at least as large as $K_2(T, K_1)$), the value $t_k + t$ can also be made large enough so that the interpolated bias term at time $t_k + t$ is arbitrarily small. Let $k \geq K_2(T, K_1)$ and

$t \in I_T$. Then, (4.20) with $|t|$ replaced by T implies:

$$\|\boldsymbol{B}_k(t)\| = \|\boldsymbol{B}_0(t_k + t) - \boldsymbol{B}_0(t_k)\| \leq \epsilon T R_3(\omega) \tag{4.21}$$

provided $\omega \in \Omega_3 \cap \Omega_4$. Since $K_1(\omega, \epsilon)$ and $K_2(T, K_1(\omega, \epsilon))$ do not depend on t and since ϵ can be taken to be as small as desired, $\boldsymbol{B}_k(t)$ converges uniformly (in t) to the zero vector for $t \in I_T$ and $\omega \in \Omega_3 \cap \Omega_4$ (note that $P(\Omega_3 \cap \Omega_4) = 1$). The fact that $\{\boldsymbol{B}_k(t)\}_{k \geq 0}$ is uniformly equicontinuous for $t \in I_T$ now follows in a similar manner to the proof of Theorem 7.24 in Rudin (1976). □

The following lemma will be useful for establishing some important properties regarding the set $\{\boldsymbol{M}_k(t)\}_{k \geq 0}$.

Lemma 4 (Kushner and Clark 1978, Lemma 2.2.1)**.** Let $\boldsymbol{\rho}_k$ be a vector-valued random variable. For $t_k = \sum_{r=0}^{k-1} a_r$ define $\boldsymbol{R}(t_k) \equiv \sum_{r=0}^{k-1} a_r \boldsymbol{\rho}_r$. For $t \in [0, \infty)$ let $\boldsymbol{R}(t)$ be the piecewise linear interpolation of $\{\boldsymbol{R}(t_k)\}_{k \geq 0}$. Additionally, for $t \in (-\infty, 0]$ let $\boldsymbol{R}(t) \equiv \boldsymbol{R}(t_0)$. Assume:

$$\lim_{k \to \infty} P\left(\sup_{m \geq k} \left\| \sum_{r=k}^{m} a_r \boldsymbol{\rho}_r \right\| \geq \varepsilon \right) = 0.$$

Then, $\boldsymbol{R}(t)$ is uniformly continuous over $t \in \mathbb{R}$ w.p.1 and $\sum_{k=0}^{\infty} a_k \boldsymbol{\rho}_k < \infty$ w.p.1.

The statement of the following Lemma is similar to the statement of Lemma 3 with the exception that it pertains to $\{\boldsymbol{M}_k(t)\}_{k \geq 0}$. The proof of the following

lemma, however, is fundamentally different from the proof of Lemma 3.

Lemma 5. Assume conditions A0 and A6 hold and for any finite $T > 0$ let I_T be defined as in Lemma 3. Then, the following statements hold for all ω in a set w.p.1 and any finite $T > 0$. The set $\{\boldsymbol{M}_k(t)\}_{k\geq 0}$ is a set of functions each of which is uniformly continuous over I_T. Additionally, the functions in $\{\boldsymbol{M}_k(t)\}_{k\geq 0}$ are bounded over $t \in I_T$ uniformly in k. Furthermore, $\boldsymbol{M}_k(t) \to \mathbf{0}$ uniformly over $t \in I_T$ as $k \to \infty$. The former statements imply $\{\boldsymbol{M}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous over I_T.

Proof. By letting $\boldsymbol{\rho}_r = \boldsymbol{D}_r$ and $\boldsymbol{R}(t) = \boldsymbol{M}_0(t)$ in Lemma 4, condition A6 implies that $\boldsymbol{M}_0(t)$ is uniformly continuous for $t \in \mathbb{R}$. Moreover, due to the manner in which $\boldsymbol{M}_k(t)$ can be obtained from $\boldsymbol{M}_0(t)$ (i.e., via shifting) we obtain the uniform continuity of $\boldsymbol{M}_k(t)$ for $t \in \mathbb{R}$ and, therefore, over $t \in I_T$. The fact that $\{\boldsymbol{M}_k(t)\}_{k\geq 0}$ is bounded over $t \in I_T$ uniformly in k also follows from Lemma 4 as is shown next.

First, using the definition of $\bar{\boldsymbol{M}}_0(t_k)$ (see (4.7b)) along with the definition of $\boldsymbol{M}_k(t)$ (see (4.12a)) it follows that for any two nonnegative integers n_1 and n_2 satisfying $n_2 \geq n_1 + 1$ the following holds:

$$\boldsymbol{M}_0(t_{n_2}) - \boldsymbol{M}_0(t_{n_1}) = \bar{\boldsymbol{M}}_0(t_{n_2}) - \bar{\boldsymbol{M}}_0(t_{n_1}) = \sum_{r=n_1}^{n_2-1} a_r \boldsymbol{D}_r$$

(recall that $\boldsymbol{D}_r$ was defined in condition A6). Next, condition A6, Lemma 4, and

the Cauchy criterion for convergence (Rudin 1976, Theorem 3.11) imply that for any $\epsilon > 0$ there exists a constant $K_1(\omega, \epsilon)$ such that for $n_1, n_2 \geq K_1(\omega, \epsilon)$:

$$\Big\| \sum_{r=n_1}^{n_2-1} a_r \boldsymbol{D}_r \Big\| < \epsilon \tag{4.22}$$

for all ω in a set of probability one. Therefore, $\|\boldsymbol{M}_0(t_{n_2}) - \boldsymbol{M}_0(t_{n_1})\| < \epsilon$. In general, for each $t \in I_T$ it follows from condition A0 (using the fact that $\sum_{k=0}^{\infty} a_k = \infty$) that there exists a constant $K_2(\omega, \epsilon, T)$ such that if $k \geq K_2(\omega, \epsilon, T)$ then:

$$t_n \leq t_k + t \leq t_{n+1} \tag{4.23}$$

for some $n \geq K_1(\omega, \epsilon)$ (the value of n depends on k and on t although this dependance has been omitted for simplicity since it does not impact our arguments). Next, note that (4.22) with $n_1 = n$ and $n_2 = n + 1$ implies that $\|\boldsymbol{M}_0(t_{n+1}) - \boldsymbol{M}_0(t_n)\| < \epsilon$. Combining this with (4.23) it follows that $\|\boldsymbol{M}_k(t)\| = \|\boldsymbol{M}_0(t_k + t) - \boldsymbol{M}_0(t_n)\| < \epsilon$ whenever $t \in I_T$ and $k \geq K_2(\omega, \epsilon, T)$. Since $K_2(\omega, \epsilon, T)$ does not depend on t, $\boldsymbol{M}_k(t)$ must be bounded over $t \in I_T$ uniformly in k. Furthermore, since ϵ can be arbitrarily small we also obtain the desired convergence of $\boldsymbol{M}_k(t) \to \mathbf{0}$ uniformly over $t \in I_T$. The uniform equicontinuity of $\{\boldsymbol{M}_k(t)\}_{k \geq 0}$ follows immediately (see the end of the proof of Lemma 3). □

The statement of the following lemma is similar to the statements of Lem-

mas 3 and 5 with the exception that it pertains to the set $\{\boldsymbol{N}_k(t)\}_{k\geq 0}$. Part of the proof of the following lemma is a combination of the proofs of Lemmas 3 and 5 and, consequently, some details have been omitted.

Lemma 6. Assume conditions A0–A4 hold and for any finite $T > 0$ let I_T be defined as in Lemma 3. Then, the following statements hold for all ω in a set w.p.1 and any finite $T > 0$. The set $\{\boldsymbol{N}_k(t)\}_{k\geq 0}$ is a set of functions each of which is uniformly continuous over I_T. Additionally, the functions in $\{\boldsymbol{N}_k(t)\}_{k\geq 0}$ are bounded over $t \in I_T$ uniformly in k. Furthermore, $\boldsymbol{N}_k(t) \to \mathbf{0}$ uniformly over $t \in I_T$ as $k \to \infty$. The former statements imply $\{\boldsymbol{N}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous over I_T.

Proof. First, rewrite the expression for $\bar{\boldsymbol{N}}_0(t_k)$ given in (4.7c) as follows:

$$\bar{\boldsymbol{N}}_0(t_k) = \sum_{r=0}^{k-1} a_r \left[\sum_{j=1}^{d} (x_r(j) - \mu_r(j) + \mu_r(j) - \mu(j))\boldsymbol{g}^{(j)}(\hat{\theta}_r)\right] = \sum_{r=0}^{k-1} a_r \boldsymbol{\nu}_r, \quad (4.24)$$

where $\boldsymbol{\nu}_r \equiv \sum_{j=1}^{d} (x_r(j) - \mu_r(j) + \mu_r(j) - \mu(j))\boldsymbol{g}^{(j)}(\hat{\theta}_r)$ with $x_r(j)$, $\mu_r(j)$, and $\mu(j)$ defined in on pp. 52–53. Using A0 (the fact that $\tilde{a}_k^{(j)}/a_k \to r_j$ w.p.1), A1, A2, A3, and A4, equation (4.24) implies that $\boldsymbol{\nu}_r$ is bounded in magnitude w.p.1 (for $\omega \in \Omega_0 \cap \Omega_1$) uniformly over r. Therefore, w.p.1 there exists a constant $R_4(\omega)$ satisfying $\|\boldsymbol{\nu}_r\| \leq R_4(\omega)$ for ω in a set w.p.1. Then, via a derivation similar to that of (4.19) it is possible to show that $\|\boldsymbol{N}_k(t)\| = \|\boldsymbol{N}_0(t_k + t) - \boldsymbol{N}_0(t_k)\| \leq |t|R_4(\omega)$. This implies $\{\boldsymbol{N}_k(t)\}_{k\geq 0}$ is bounded over $t \in I_T$ uniformly in k.

Because $\boldsymbol{N}_k(t)$ is a continuous function on I_T it is clear that $\boldsymbol{N}_k(t)$ must also be uniformly continuous on I_T. It remains to show that $\boldsymbol{N}_k(t)$ converges to the zero vector uniformly over $t \in I_T$. We begin by rewriting the expression for the vector $\boldsymbol{\nu}_r$ appearing in (4.24). First, using the last part of condition A2:

$$x_r(j) - \mu_r(j) = E\left[\frac{\tilde{a}_r^{(j)}}{a_r}\right](C_r(j) - E[C_r(j)]) + \left(\frac{\tilde{a}_r^{(j)}}{a_r} - E\left[\frac{\tilde{a}_r^{(j)}}{a_r}\right]\right) C_r(j). \tag{4.25}$$

Using (4.25) let us express $\boldsymbol{\nu}_r$ as $\boldsymbol{\nu}_r = \boldsymbol{\nu}_r' + \boldsymbol{\nu}_r''$, where

$$\boldsymbol{\nu}_r' \equiv \sum_{j=1}^{d} E\left[\frac{\tilde{a}_r^{(j)}}{a_r}\right](C_r(j) - E[C_r(j)])\boldsymbol{g}^{(j)}(\hat{\boldsymbol{\theta}}_r),$$
$$\boldsymbol{\nu}_r'' \equiv \sum_{j=1}^{d} \left[\eta_r(j)C_r(j) + \mu_r(j) - \mu(j)\right]\boldsymbol{g}^{(j)}(\hat{\boldsymbol{\theta}}_r),$$

and $\eta_k(j) \equiv \tilde{a}_k^{(j)}/a_k - E[\tilde{a}_k^{(j)}/a_k]$. Note that A0 implies $\eta_k(j) \to 0$ w.p.1. Therefore, since $C_k(j)$ is bounded (a consequence of condition A1) it follows that $\eta_k(j)C_k(j) \to 0$ w.p.1. Next we study the asymptotic properties of $\boldsymbol{\nu}_r'$ and $\boldsymbol{\nu}_r''$.

Conditions A2, A3, and A4 along with the fact that $\eta_k(j)C_k(j) \to 0$ w.p.1 imply $\boldsymbol{\nu}_r'' \to \boldsymbol{0}$ w.p.1. The term $\boldsymbol{\nu}_r'$, on the other hand, cannot be assumed to converge. However, it is possible to show $\sum_{k=0}^{\infty} a_k \boldsymbol{\nu}_k' < \infty$, which we do next. Using the definition of S_r given in condition A2, conditions A0 and A1 imply that for a fixed k the sequence $\{\sum_{r=k}^{m} a_r S_r\}_{m \geq k}$ (indexed by m) is a martingale

sequence. An implication of p. 315 in Doob (1953) is the following:

$$P\left(\sup_{m\geq k}\left\|\sum_{r=k}^{m}a_rS_r\right\|\geq\varepsilon\right)\leq\varepsilon^{-2}E\left\|\sum_{r=k}^{\infty}a_rS_r\right\|^2=\varepsilon^{-2}\sum_{r=k}^{\infty}a_r^2\operatorname{Var}(S_r),\tag{4.26}$$

where equality holds by condition A2 (the fact that $E[S_kS_\ell]=0$ for $k\neq\ell$). Moreover, since the $C_k(j)$ are bounded uniformly in k and j (condition A1) and since the sequence $E[\tilde{a}_k^{(j)}/a_k]$ converges (condition A0), the sequence S_k must be bounded uniformly in k. Therefore, there exists an $\sigma^2>0$ such that $\operatorname{Var}(S_k)\leq\sigma^2$ for all k. Then, (4.26) along with condition A0 implies:

$$\lim_{k\to\infty}P\left(\sup_{m\geq k}\left\|\sum_{r=k}^{m}a_rS_r\right\|\geq\varepsilon\right)\leq\lim_{k\to\infty}\frac{\sigma^2}{\varepsilon^2}\sum_{r=k}^{\infty}a_r^2=0.$$

Then, Lemma 4 implies:

$$\sum_{k=0}^{\infty}a_k\sum_{j=1}^{d}E\left[\frac{\tilde{a}_k^{(j)}}{a_k}\right](C_k(j)-E[C_k(j)])=\sum_{k=0}^{\infty}a_kS_k<\infty\text{ w.p.1.}$$

Furthermore, since $\|\boldsymbol{g}^{(j)}(\hat{\theta}_k)\|$ is bounded w.p.1 (conditions A3 and A4):

$$\sum_{k=0}^{\infty}a_k\sum_{j=1}^{d}E\left[\frac{\tilde{a}_k^{(j)}}{a_k}\right](C_k(j)-E[C_k(j)])\boldsymbol{g}^{(j)}(\hat{\theta}_k)=\sum_{k=0}^{\infty}a_k\boldsymbol{\nu}_k'<\infty.\tag{4.27}$$

Following the ideas in the proofs of Lemmas 3 and 5, equation (4.27) and the fact that $\boldsymbol{\nu}_k''\to 0$ w.p.1 can be used to show that $\boldsymbol{N}_k(t)$ converges to the zero

vector uniformly over $t \in I_T$. Specifically, the term $\boldsymbol{\nu}_r'$ can be related to the term $\boldsymbol{D}_r$ appearing in condition A6 and $\boldsymbol{\nu}_r''$ can be related to the term:

$$\sum_{m=1}^{s_r} \sum_{i=0}^{n_r(m)-1} \left(\frac{\tilde{A}_r(m)}{a_r} \right) \left\| \boldsymbol{\beta}_r^{(j_r(m))} \left(\hat{\boldsymbol{\theta}}_r^{(I_{m,i})} \right) \right\|$$

appearing in (4.17). Once again, the uniform equicontinuity of $\{\boldsymbol{N}_k(t)\}_{k\geq 0}$ follows immediately (see the proof of Lemma 3). □

The statement of the following lemma is similar to the statements of Lemmas 3, 5, and 6, with the exception that it pertains to the set $\{\boldsymbol{W}_k(t)\}_{k\geq 0}$.

Lemma 7. Assume conditions A0, A1, and A3–A6 hold and for any finite $T > 0$ let I_T be defined as in Lemma 3. Then, the following statements hold for all ω in a set w.p.1 and any finite $T > 0$. The set $\{\boldsymbol{W}_k(t)\}_{k\geq 0}$ is a set of functions each of which is uniformly continuous over I_T. Additionally, the functions in $\{\boldsymbol{W}_k(t)\}_{k\geq 0}$ are bounded over $t \in I_T$ uniformly in k. Furthermore, $\boldsymbol{W}_k(t) \to \boldsymbol{0}$ uniformly over $t \in I_T$ as $k \to \infty$. The former statements imply $\{\boldsymbol{W}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous over I_T.

Proof. First we prove the boundedness part of the lemma. Recall the definition of $\bar{\boldsymbol{W}}_0(t_k)$ in (4.7d) (we include it here for convenience):

$$\bar{\boldsymbol{W}}_0(t_k) = \sum_{r=0}^{k-1} a_r \sum_{m=1}^{s_r} \sum_{i=0}^{n_r(m)-1} \left(\frac{\tilde{A}_r(m)}{a_r} \right) \left[\boldsymbol{g}^{(j_r(m))} \left(\hat{\boldsymbol{\theta}}_r^{(I_{m,i})} \right) - \boldsymbol{g}^{(j_r(m))}(\hat{\boldsymbol{\theta}}_r) \right]. \quad (4.28)$$

Conditions A3 and A4 imply all terms of the form $\boldsymbol{g}^{(j)}(\cdot)$ in (4.28) are bounded in magnitude by a constant $R_5(\omega)$ for all $\omega \in \Omega_1$. Thus, using Lemma 1 and following a derivation similar to that of (4.19) we obtain that $\|\boldsymbol{W}_k(t)\| = \|\boldsymbol{W}_0(t_k + t) - \boldsymbol{W}_0(t_k)\| \leq |t| R_3(\omega) R_5(\omega)$. For $t \in I_T$ this implies $\|\boldsymbol{W}_k(t)\| \leq T R_3(\omega) R_5(\omega)$. Thus, $\{\boldsymbol{W}_k(t)\}_{k \geq 0}$ is bounded uniformly in k for $t \in I_T$. The uniform continuity of each $\boldsymbol{W}_k(t)$ function on finite (i.e., closed and bounded) intervals follows by the continuity of $\boldsymbol{W}_k(t)$. Next we show uniform convergence of $\boldsymbol{W}_k(t)$ to zero.

First recall that $\hat{\theta}_k^{(I_{m,i})}$ satisfies:

$$\hat{\theta}_k^{(I_{m,i})} = \hat{\theta}_k - \sum_{z=1}^{m-1} \sum_{\ell=0}^{n_k(z)-1} \left[\tilde{A}_k(z)\, \hat{\boldsymbol{g}}_k^{(j_k(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) \right] - \sum_{\ell=0}^{i-1} \left[\tilde{A}_k(m)\, \hat{\boldsymbol{g}}_k^{(j_k(m))} \left(\hat{\theta}_k^{(I_{m,\ell})} \right) \right] \tag{4.29}$$

(Line 7 of Algorithm 1). Next, recall that $\hat{\boldsymbol{g}}_k^{(j_k(z))}(\hat{\theta}_k^{(I_{z,\ell})})$ can be written as the sum $\boldsymbol{g}^{(j_k(z))}(\hat{\theta}_k^{(I_{z,\ell})}) + \beta_k^{(j_k(z))}(\hat{\theta}_k^{(I_{z,\ell})}) + \xi_k^{(j_k(z))}(\hat{\theta}_k^{(I_{z,\ell})})$. Furthermore, $\boldsymbol{g}^{(j_k(z))}(\hat{\theta}_k^{(I_{z,\ell})})$ and $\beta_k^{(j_k(z))}(\hat{\theta}_k^{(I_{z,\ell})})$ are bounded w.p.1 (conditions A3–A5). Therefore, using condition A0 (the implication that $\tilde{a}_k^{(j)} \to 0$ w.p.1 for all j) and condition A1 we rewrite (4.29) as follows:

$$\hat{\theta}_k^{(I_{m,i})} = \hat{\theta}_k - \sum_{z=1}^{m-1} \sum_{\ell=0}^{n_k(z)-1} \tilde{A}_k(z)\, \xi_k^{(j_k(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) - \sum_{\ell=0}^{i-1} \tilde{A}_k(m)\, \xi_k^{(j_k(m))} \left(\hat{\theta}_k^{(I_{m,\ell})} \right) + \boldsymbol{T}_k, \tag{4.30}$$

where $\boldsymbol{T}_k$ is a vector such that $\boldsymbol{T}_k \to 0$ w.p.1. Additionally, condition A6 and

Lemma 4 imply $\sum_{k=0}^{\infty} a_k \boldsymbol{D}_k < \infty$ w.p.1. Therefore,

$$\sum_{z=1}^{m-1} \sum_{\ell=0}^{n_k(z)-1} \left[\tilde{A}_k(z)\ \xi_k^{(j_k(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) \right] + \sum_{\ell=0}^{i-1} \left[\tilde{A}_k(m)\ \xi_k^{(j_k(m))} \left(\hat{\theta}_k^{(I_{m,\ell})} \right) \right] \to \mathbf{0} \text{ w.p.1} \tag{4.31}$$

(this is a consequence of Theorem 3.11 in Rudin 1976, also known as the Cauchy criterion for convergence). Using (4.30) and (4.31) and we can write $\hat{\theta}_k^{(I_{m,i})} = \hat{\theta}_k + \boldsymbol{T}_k'$, where $\boldsymbol{T}_k'$ is a vector such that $\boldsymbol{T}_k' \to \mathbf{0}$ w.p.1 as $k \to \infty$. Consequently, condition A3 implies $\|\boldsymbol{g}^{(j_r(m))}(\hat{\theta}_r^{(I_{m,i})}) - \boldsymbol{g}^{(j_r(m))}(\hat{\theta}_r)\| \to 0$ w.p.1 as $r \to \infty$ uniformly in m and i. Then, via a derivation similar to that of (4.21) it follows that $\|\boldsymbol{W}_0(t_k + t) - \boldsymbol{W}_0(t_k)\| \leq T R_3(\omega)\epsilon$ for k large enough and $t \in I_T$. Uniform equicontinuity then follows (see the end of the proof of Lemma 3). □

Lemma 8. Assume conditions A2–A4 hold and for any finite $T > 0$ let I_T be defined as in Lemma 3. Then, the following statements hold for all ω in a set w.p.1 and any finite $T > 0$. The set of functions $\{\int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\ ds\}_{k \geq 0}$ is bounded over $t \in I_T$ uniformly in k. Furthermore, $\{\int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\ ds\}_{k \geq 0}$ is uniformly equicontinuous over I_T.

Proof. Recall from (4.5) that $\boldsymbol{h}(\theta) = \sum_{j=1}^{d} \mu(j) \boldsymbol{g}^{(j)}(\theta)$. Then,

$$\int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\ ds = \int_0^t \sum_{j=1}^{d} \mu(j) \boldsymbol{g}^{(j)}(\bar{\boldsymbol{Z}}_k(s))\ ds. \tag{4.32}$$

Since $\boldsymbol{g}^{(j)}(\bar{\boldsymbol{Z}}_k(t))$ is bounded w.p.1 uniformly over k and t (conditions A3 and

A4) and since $\mu(j)$ must be bounded uniformly in j (condition A2), then the magnitude of the integrand in (4.32) must be bounded uniformly over t by some constant $R_6(\omega)$ for ω in a set w.p.1. Thus, the boundedness part of the lemma follows for $t \in I_T$. Next, for $t_1, t_2 \in I_T$ with $t_1 < t_2$:

$$\left\| \int_0^{t_2} \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\, ds - \int_0^{t_1} \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\, ds \right\| \leq \int_{t_1}^{t_2} \left\| \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s)) \right\|\, ds \leq |t_2 - t_2| R_6(\omega)$$

independently of k. Therefore, the desired equicontinuity also follows. □

The following Lemma presents a result similar to that of Lemmas 2–8 for the set of functions $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ with one important modification: parts of the result are now shown to hold for all $t \in \mathbb{R}$ rather than only for $t \in I_T$.

Lemma 9. Assume conditions A0–A6 hold. Then, the functions in the set $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ are bounded over $t \in \mathbb{R}$ uniformly in k w.p.1. Additionally, $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous for $t \in \mathbb{R}$ w.p.1.

Proof. The desired uniform boundedness was proven in Lemma 2. Next, recall from (4.13) that:

$$\boldsymbol{Z}_k(t) = \boldsymbol{Z}_k(0) - \boldsymbol{B}_k(t) - \boldsymbol{M}_k(t) - \boldsymbol{N}_k(t) - \boldsymbol{W}_k(t) - \int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\, ds.$$

Here, Lemmas 3–8 and the fact that $\boldsymbol{Z}_k(t)$ is a shifted version of $\boldsymbol{Z}_0(t)$ imply that $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ is a set of uniformly equicontinuous functions for $t \in \mathbb{R}$. □

Lemma 10. Assume conditions A0–A6 hold and for any finite $T > 0$ let I_T be defined as in Lemma 3. Then, for any finite $T > 0$ it follows that $\boldsymbol{\zeta}_k(t) \to \mathbf{0}$ (recall $\boldsymbol{\zeta}_k(t)$ was defined in (4.14)) w.p.1 uniformly over $t \in I_T$ as $k \to \infty$.

Proof. We remind the reader that any unspecified probabilistic arguments are meant w.p.1. The vector $\boldsymbol{\zeta}_k(t)$ can be written as:

$$\begin{aligned}\boldsymbol{\zeta}_k(t) &= \int_0^t \boldsymbol{h}(\boldsymbol{Z}_k(s))\, ds - \int_0^t \boldsymbol{h}(\bar{\boldsymbol{Z}}_k(s))\, ds \\ &= \int_0^t \sum_{j=1}^{d} \mu(j) \left[\boldsymbol{g}^{(j)}(\boldsymbol{Z}_k(s)) - \boldsymbol{g}^{(j)}(\bar{\boldsymbol{Z}}_k(s))\right]\, ds. \end{aligned} \tag{4.33}$$

By condition A4, the vectors $\bar{\boldsymbol{Z}}_k(s)$ and $\boldsymbol{Z}_k(s)$ must lie in a p-dimensional ball of radius $R_1(\omega)$ for all $\omega \in \Omega_1$. Let $\Theta(\omega) \subset \mathbb{R}^p$ denote the closure (using the Euclidean topology) of the aforementioned ball. Then, condition A3 implies $\boldsymbol{g}(\theta)$ is uniformly continuous over $\theta \in \Theta(\omega)$. In other words, for any $\epsilon > 0$ there exists a $\delta_1(\omega, \epsilon) > 0$ such that $\|\boldsymbol{g}^{(j)}(\boldsymbol{Z}_k(s)) - \boldsymbol{g}^{(j)}(\bar{\boldsymbol{Z}}_k(s))\| < \epsilon$ if $\|\bar{\boldsymbol{Z}}_k(s) - \boldsymbol{Z}_k(s)\| < \delta_1(\omega, \epsilon)$. Next, Lemma 9 implies $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous w.p.1 on I_T. Therefore, for $t_1, t_2 \in I_T$ and any $\epsilon > 0$, there exists a $\delta_2(\epsilon, T, \omega) > 0$ such that $\|\boldsymbol{Z}_k(t_2) - \boldsymbol{Z}_k(t_1)\| < \delta_1(\omega, \epsilon)$ whenever $|t_2 - t_1| < \delta_2(\epsilon, T, \omega)$ independently of k. Next, note that for any $\delta_2(\epsilon, T, \omega) > 0$ and $s \in I_T$ it is possible to take k large enough (independently of t) so that $t_m \leq t_k + s \leq t_{m+1}$ where $t_{m+1} - t_m = a_m < \delta_2(\epsilon, T, \omega)$ (this follows from condition A0). Then, for large enough k and $s \in I_T$ we have $\|\boldsymbol{Z}_k(s) - \bar{\boldsymbol{Z}}_k(s)\| \leq \|\boldsymbol{Z}_k(t_{m+1}) - \boldsymbol{Z}_k(t_m)\| < \delta_1(\epsilon, \omega)$

and, therefore, $\|\boldsymbol{g}^{(j)}(\boldsymbol{Z}_k(s)) - \boldsymbol{g}^{(j)}(\bar{\boldsymbol{Z}}_k(s))\| < \epsilon$ for all $s \in I_T$. Consequently, the maximum possible norm of the integral in (4.33) for $t \in I_T$ is ϵT, where $\epsilon > 0$ can be as small as desired provided k is large (precisely how large is independent of t). Thus, we have proven the desired uniform (in t) convergence to the zero vector. □

4.3 A Convergence Theorem for GCSA

We are now ready to prove the following theorem for convergence of the GCSA algorithm. Once again, all unspecified probabilistic arguments in the statement of the theorem as well as in its proof are meant w.p.1.

Theorem 2. Let $\hat{\theta}_k$ denote the GCSA iterates of Algorithm 1 and assume conditions A0–A6 hold. Then, there exists a subsequence $\{\boldsymbol{Z}_{k_i}(t)\}_{i\geq 1}$ of $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ and a bounded (w.p.1) function $\boldsymbol{Z}(t)$ such that $\boldsymbol{Z}_{k_i}(t) \to \boldsymbol{Z}(t)$ uniformly over $t \in I_T$ for any finite $T > 0$ as $i \to \infty$ (I_T was defined in Lemma 3). The aforementioned limit function, $\boldsymbol{Z}(t)$, satisfies the ODE in (4.15). If in addition conditions A7 and A8 hold (i.e., if A0–A8 hold) then $\hat{\theta}_k \to \theta^*$ w.p.1.

Proof. The proof of this theorem consists of three main steps:

Step 1: Show that A0–A6 imply that for $t \in \mathbb{R}$ the functions $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ are bounded uniformly in k, and $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ is uniformly equicontinuous.

Step 2: Given Step 1, the Arzelà–Ascoli Theorem implies that the sequence $\{\boldsymbol{Z}_k(t)\}_{k\geq 0}$ must have a subsequence $\{\boldsymbol{Z}_{k_i}(t)\}_{i\geq 1}$ that converges to some bounded function $\boldsymbol{Z}(t)$ uniformly over $t \in I_T$ for any finite $T > 0$ as $i \to \infty$. This step consists of showing that $\boldsymbol{Z}(t)$ must satisfy the ODE in condition A7.

Step 3: Using the asymptotic stability condition (condition A7) along with A8 show that $\hat{\theta}_k$ must converge to θ^*.

Step 1 was completed in Lemma 9. Step 2 is an implication of Lemmas 3–7, 9, and 10 as is shown next. Using the Arzelà–Ascoli Theorem (Kushner and Clark 1978, p. 20) along with the fact that $\boldsymbol{Z}_k(t)$ is a shifted version of $\boldsymbol{Z}_0(t)$ it follows that there exists a subsequence $\{\boldsymbol{Z}_{k_i}(t)\}_{i\geq 1}$ and a bounded (w.p.1) function $\boldsymbol{Z}(t)$ such that $\boldsymbol{Z}_{k_i}(t) \to \boldsymbol{Z}(t)$ uniformly over $t \in I_T$ for any finite $T > 0$ as $i \to \infty$. Then, by (4.14) in conjunction with Lemmas 3–7 and 10 (using the convergence to zero of the different functions) it follows that $\boldsymbol{Z}(t)$ satisfies:

$$\boldsymbol{Z}(t) = \lim_{i\to\infty} \boldsymbol{Z}_{k_i}(t) = \boldsymbol{Z}(0) - \lim_{i\to\infty} \int_0^t \boldsymbol{h}(\boldsymbol{Z}_{k_i}(s))\, ds,$$

for $t \in I_T$. Due to the uniform convergence of $\boldsymbol{Z}_{k_i}(t)$ on I_T and using the continuity of $\boldsymbol{g}(\theta)$, it follows that:

$$\boldsymbol{h}(\boldsymbol{Z}_{k_i}(t)) \to \sum_{j=1}^{d} \mu(j)\boldsymbol{g}^{(j)}(\boldsymbol{Z}(t)) = \boldsymbol{h}(\boldsymbol{Z}(t)) \text{ w.p.1} \tag{4.34}$$

uniformly over $t \in I_T$ as $i \to \infty$. Using the uniform convergence in (4.34) along with Lebesgue's dominated convergence theorem implies:

$$\boldsymbol{Z}(t) = \boldsymbol{Z}(0) - \int_0^t \boldsymbol{h}(\boldsymbol{Z}(s))\, ds \tag{4.35}$$

for $t \in I_T$. Because $T > 0$ is arbitrary, (4.35) holds for all t. Therefore,

$$\dot{\boldsymbol{Z}}(t) = -\sum_{j=1}^{d} \mu(j) \boldsymbol{g}^{(j)}(\boldsymbol{Z}(t)),$$

which is the ODE in (4.15). The remainder of the proof (Step 3) is now identical to the last part of the proof of Theorem 2.3.1 in Kushner and Clark (1978). Therefore, only a brief outline of Step 3 is included below.

By condition A8, we know there exists a subsequence $\{\hat{\theta}_{k_i}\}_{i \geq 1}$ such that $\hat{\theta}_{k_i} \in A$ for all $i \geq 1$. Furthermore, by Step 2 we can assume without loss of generality that $\boldsymbol{Z}_{k_i}(t) \to \boldsymbol{Z}(t)$ where $\boldsymbol{Z}(t)$ satisfies the ODE in condition A7. Additionally, since $\boldsymbol{Z}_{k_i}(0) = \hat{\theta}_{k_i} \in A$ where A is a compact set then $\boldsymbol{Z}(0) \in A$ by construction. Because θ^* is an assymptotically stable (in the sense of Lyapunov) solution to the differential equation (4.15), we know $\lim_{t \to \infty} \boldsymbol{Z}(t) = \theta^*$. The stability properties of θ^* then guarantee $\hat{\theta}_k \to \theta^*$ (see Kushner and Clark 1978, pp. 42–43). □

Corollary 1. If the conditions of Theorem 2 hold, then the sequence $\{\hat{\theta}_k^{(I_{m,i})}\}$ of Algorithm 1 converges to θ^* w.p.1 uniformly in m and i as $k \to \infty$.

Proof. This result follows immediately from Theorem 2 along with the comment below (4.31). □

The following section obtains Corollaries to Theorem 2 pertaining to two special cases of the GCSA algorithm.

4.4 Two Special Cases of GCSA

Theorem 2 gave convergence conditions for the GCSA algorithm, two special cases of which were discussed in Section 3.4 (see (3.14) and (3.18)). In (3.18), for example, the parameter vector was partitioned into two subvectors; at each iteration the subvector to update was chosen at random. A generalization of (3.18) is obtained by partitioning θ into d subvectors. The resulting generalization of (3.18) is presented in Algorithm 2 below. For simplicity, it is assumed the sets $\mathcal{S}_1, \ldots, \mathcal{S}_d$ in (3.19) satisfy:

$$\bigcup_{i=1}^{d} \mathcal{S}_i = \mathcal{S}, \quad \bigcap_{i=1}^{d} \mathcal{S}_i = \emptyset. \tag{4.36}$$

In order to study the behavior of Algorithm 2 we first prove two lemmas that are useful for understanding the behavior of the sequence $\tilde{a}_k^{(j)}/a_k$, a sequence whose relevance stems from condition A0 in Theorem 2 (also note that this ratio affects the value of $\boldsymbol{h}(\theta)$ through (4.4)). The following lemma gives sufficient conditions for the convergence w.p.1 of $\tilde{a}_k^{(j)}/a_k$.

Algorithm 2 Randomized Subvector Selection

Require: $\hat{\theta}_0$, $\{a_k^{(j)}\}_{k\geq 0}$ for $j = 1, \ldots, d$, and let $\mathcal{S}_1, \ldots, \mathcal{S}_d$ be such that (4.36) holds. Set $k = 0$.

1: **while** stopping criterion has not been reached **do**

2: Let j_k be a random variable with $P(j_k = j) = q(j) \neq 0$ and $\sum_{j=1}^{d} q(j) = 1$.

3: Define:

$$\hat{\theta}_{k+1} \equiv \hat{\theta}_k - \tilde{a}_k^{(j_k)} \hat{\boldsymbol{g}}^{(j_k)}(\hat{\theta}_k).$$

4: set $k = k + 1$

5: **end while**

Lemma 11. Consider the setting of Algorithm 2. Assume $a_k > 0$ and that $a_k^{(j)} > 0$ is a monotonically decreasing sequence with $a_k^{(j)} \to 0$. In addition, assume there exists a function $\rho(j, q)$ with domain $\{1, \ldots, j\} \times [0, 1]$ and range in $\mathbb{R}$ such that if $\epsilon > 0$ is small enough that $0 \leq q(j) \pm \epsilon \leq 1$ then:

$$\lim_{k\to\infty} \frac{a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}}{a_k} = \rho(j, q(j) - \epsilon), \quad \lim_{k\to\infty} \frac{a^{(j)}_{\lceil k(q(j)+\epsilon)\rceil}}{a_k} = \rho(j, q(j) + \epsilon),$$

($\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the floor and ceiling functions, respectively). Assume there exist a constant $C < \infty$ such that $0 < \rho(j, q(j) \pm \epsilon) < C$. Finally, let:

$$\lim_{\epsilon\to 0} \rho(j, q(j) \pm \epsilon) = \rho(j, q(j)) \in [\rho(j, q(j) + \epsilon), \rho(j, q(j) - \epsilon)].$$

Then, $\tilde{a}_k^{(j)}/a_k \to \rho(j, q(j))$ w.p.1 with $0 < \rho(j, q(j)) < C$.

Proof. Throughout this proof all unspecified probabilistic arguments are meant to hold w.p.1. From (3.22) we know that for $k > 0$:

$$\tilde{a}_{k+1}^{(j)} = a_0^{(j)} + \sum_{i=0}^{k} \chi\left\{ \frac{\varphi_k^{(j)}}{k} - q(j) \geq \frac{i}{k} - q(j) \right\} (a_{i+1}^{(j)} - a_i^{(j)}), \tag{4.37}$$

where $0 \leq \varphi_k^{(j)} \leq k$ was defined in (3.21). Furthermore, the strong law of large numbers implies that $\varphi_k^{(j)}/k - q(j) \to 0$ w.p.1 as $k \to \infty$. This implies (w.p.1) that for all $\epsilon > 0$ there exists an $N(\epsilon)$ such that $|\varphi_k^{(j)}/k - q(j)| < \epsilon$ for $k \geq N(\epsilon)$ (the constant $N(\epsilon)$ depends on ω although the dependence has been omitted for simplicity). Therefore,

$$\chi\left\{ \frac{\varphi_k^{(j)}}{k} - q(j) \geq \frac{i}{k} - q(j) \right\} = \begin{cases} 0 & \text{if } k \geq N(\epsilon) \text{ and } i \geq \lceil k(\epsilon + q(j)) \rceil, \\ 1 & \text{if } k \geq N(\epsilon) \text{ and } i \leq \lfloor k(q(j) - \epsilon) \rfloor. \end{cases} \tag{4.38}$$

Equation (4.38) implies $\tilde{a}_{k+1}^{(j)} \in [a^{(j)}_{\lceil k(\epsilon+q(j)) \rceil}, a^{(j)}_{\lfloor k(q(j)-\epsilon) \rfloor + 1}]$ for $k \geq N(\epsilon)$. Therefore:

$$\tilde{a}_k^{(j)} \in \left[a^{(j)}_{\lceil k(q(j)+\epsilon) - (q(j)+\epsilon) \rceil}, a^{(j)}_{\lfloor k(q(j)-\epsilon) - (q(j)-\epsilon) \rfloor + 1} \right] \tag{4.39}$$

for k large. From (4.39) it follows that for large k and $\epsilon > 0$ small enough that $0 \leq q(j) \pm \epsilon \leq 1$:

$$\frac{\tilde{a}_k^{(j)}}{a_k} \in \left[\frac{a^{(j)}_{\lceil k(q(j)+\epsilon) \rceil}}{a_k}, \frac{a^{(j)}_{\lfloor k(q(j)-\epsilon) \rfloor}}{a_k} \right].$$

Therefore, $\tilde{a}_k^{(j)}/a_k$ must approach the interval $I(\epsilon) \equiv [\rho(j, q(j) + \epsilon), \rho(j, q(j) - \epsilon)]$ as k grows (i.e., the distance between $\tilde{a}_k^{(j)}/a_k$ and its closest point in the interval $I(\epsilon)$ approaches zero). Because $I(\epsilon)$ contains the point $\rho(j, q(j))$ for all $\epsilon > 0$ small enough (by assumption) and because the length of $I(\epsilon)$ decreases as ϵ decreases, the sequence $\tilde{a}_k^{(j)}/a_k$ must converge w.p.1 to $0 < \rho(j, q(j)) < C$. □

Let us comment on the assumptions of Lemma 11. The first requirement is that $q(j) > 0$ depends neither on k nor on m. In other words, when selecting the coordinates to update from one of the sets $\mathcal{S}_1, \ldots, \mathcal{S}_d$, we require the probability of selecting $\mathcal{S}_j$ to be independent of the iteration and block numbers. The remaining assumptions are satisfied for sequences of the form $a_k^{(j)} = a^{(j)}/(k + A^{(j)} + 1)^\alpha$ and $a_k = a/(k+1)^\alpha$ where $a^{(j)} > 0$, $a > 0$, $A^{(j)} > 0$, and $0 < \alpha$. To see this, note that $\lim_{k\to\infty} a_k^{(j)}/a_k = a^{(j)}/a$ and that:

$$\lim_{k\to\infty} \frac{a_{\lfloor ck \rfloor}^{(j)}}{a_k^{(j)}} = \lim_{k\to\infty} \left(\frac{k+1}{\lfloor ck \rfloor + A^{(j)} + 1} \right)^\alpha = \frac{1}{c^\alpha},$$

for $c > 0$. The last limit can be computed by using c_k as an upper bound for $\lfloor ck \rfloor$, using $ck - 1$ as a lower bound for $\lfloor ck \rfloor$, and observing that both the upper and lower bounds result in the same limit: $1/c^\alpha$. Therefore, $\lim_{k\to\infty} a_{\lfloor ck \rfloor}^{(j)}/a_k = a^{(j)}/ac^\alpha$. Similarly, it is possible to show $\lim_{k\to\infty} a_{\lceil ck \rceil}^{(j)}/a_k = a^{(j)}/(ac^\alpha)$. In the notation of Lemma 11: $\rho(j, q(j) - \epsilon) = a^{(j)}/[a(q(j) - \epsilon)^\alpha]$ for small $\epsilon > 0$ and $\rho(j, q(j) + \epsilon) = a^{(j)}/[a(q(j) + \epsilon)^\alpha]$. Then, all the conditions of Lemma 11 are

satisfied with $0 < \rho(j, q(j)) = a^{(j)}/(aq(j)^{\alpha}) < \infty$.

Lemma 11 gave conditions for the convergence w.p.1 of $\tilde{a}_k^{(j)}/a_k$. It is also of interest to derive conditions for the convergence of $E[\tilde{a}_k^{(j)}/a_k]$ (condition A0). For this we first define the concept of uniform integrability.

Definition 3. [Chung 2001] Given a sequence $\mathcal{X}_k$ of random variables, the set $\{\mathcal{X}_k\}_{k\geq 0}$ is said to be *uniformly integrable* if and only if:

$$\lim_{n\to\infty} \int_{|\mathcal{X}_k|>n} |\mathcal{X}_k| \, dP = 0 \text{ uniformly in } k.$$

Lemma 12. Assume $a_k^{(j)} > 0$ and $a_k > 0$ are both strictly monotonically decreasing sequences and that all the assumptions of Lemma 11 hold. Furthermore, let $\epsilon > 0$ be such that $q(j) - \epsilon > 0$ and assume $e^{-2\epsilon^2 k}/a_k \to 0$ as $k \to \infty$ (this last assumption holds when a_k satisfies condition A0). Then, $E[\tilde{a}_k^{(j)}/a_k] \to \rho(j, q(j))$.

Proof. For each $j = 1, \ldots, d$ we will show that the set $\{\tilde{a}_k^{(j)}/a_k\}_{k\geq 0}$ is uniformly integrable. Then, the result will follow from Lemma 11 along with Theorem 4.5.4 in Chung (2001), which states that $E[\tilde{a}_k^{(j)}/a_k] \to \rho(j, q(j))$ if the set $\{\tilde{a}_k^{(j)}/a_k\}_{k\geq 0}$ is uniformly integrable and $\tilde{a}_k^{(j)}/a_k \to \rho(j, q(j))$ w.p.1.

For each j define $\mathcal{X}_k^{(j)} \equiv \tilde{a}_k^{(j)}/a_k > 0$ and $n_i \equiv a_0^{(j)}/a_i$. Then, $n_i \to \infty$ as $i \to \infty$. Furthermore, for each $i \geq 0$ and any probability measure P:

$$\int_{\mathcal{X}_k^{(j)}>n_i} \mathcal{X}_k^{(j)} \, dP \leq \sup_{k\geq 0} \left\{ \int_{\mathcal{X}_k^{(j)}>n_i} \mathcal{X}_k^{(j)} \, dP \right\}. \tag{4.40}$$

Using the fact that a_k and $a_k^{(j)}$ are both monotonically-decreasing sequences of strictly positive numbers, for each k we have:

$$\int_{\mathcal{X}_k^{(j)}>n_i} \mathcal{X}_k^{(j)}\, dP \leq P(\mathcal{X}_k^{(j)} > n_i)\frac{a_0^{(j)}}{a_k}.$$

Combining this bound with (4.40):

$$\int_{\mathcal{X}_k^{(j)}>n_i} \mathcal{X}_k^{(j)}\, dP \leq \sup_{k\geq 0}\left\{P(\mathcal{X}_k^{(j)} > n_i)\frac{a_0^{(j)}}{a_k}\right\} = a_0^{(j)}\sup_{k\geq i}\left\{\frac{P(\mathcal{X}_k^{(j)} > n_i)}{a_k}\right\}. \quad (4.41)$$

Because $a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}/a_k \to \rho(j,(q(j)-\epsilon))$ (by assumption), for each $\delta > 0$ there exists an $N(\delta) > 0$ such that $a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}/a_k < \rho(j,q(j)-\epsilon)+\delta$ whenever $k \geq N(\delta)$. Let $M(\delta) > 0$ be such that $\rho(j,q(j)-\epsilon)+\delta < a_0^{(j)}/a_i$ for $i \geq M(\delta)$ ($M(\delta)$ exists because a_i is monotonically decreasing). Then, for $k \geq i \geq \max\{M(\delta), N(\delta)\}$:

$$\frac{a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}}{a_k} < \rho(j,q(j)-\epsilon)+\delta \leq \frac{a_0^{(j)}}{a_i} = n_i.$$

We then have the following bound when $k \geq i \geq \max\{M(\delta), N(\delta)\}$:

$$P(\mathcal{X}_k^{(j)} > n_i) \leq P\left(\frac{\tilde{a}_k^{(j)}}{a_k} > \frac{a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}}{a_k}\right) = P(\tilde{a}_k^{(j)} > a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}). \quad (4.42)$$

The term $P(\tilde{a}_k^{(j)} > a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor})$ can be computed exactly. We do this next.

From (4.37) we can see that $\tilde{a}_k^{(j)}$ will be strictly greater than $a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}$ if

the sequence $\{a_k^{(j)}\}_{k\geq 0}$ has been used strictly less than $\lfloor k(q(j)-\epsilon)\rfloor$ times by the time $\tilde{a}_k^{(j)}$ is computed (this is due to the strictly monotonically decreasing nature of $a_k^{(j)}$). The number of times $\{a_k^{(j)}\}_{k\geq 0}$ has been used can be modeled using a Binomial random variable with "success" probability $q(j)$. Therefore,

$$P(\tilde{a}_k^{(j)} > a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}) = \sum_{i=0}^{\lfloor k(q(j)-\epsilon)\rfloor-1} \binom{k}{i} q(j)^i (1-q(j))^{k-i}. \tag{4.43}$$

Since $\lfloor k(q(j)-\epsilon)\rfloor - 1 \leq k(q(j)-\epsilon)$, Hoeffding's inequality (e.g., Hoeffding 1963) allows us to further bound the probability in (4.43) as follows:

$$P(\tilde{a}_k^{(j)} > a^{(j)}_{\lfloor k(q(j)-\epsilon)\rfloor}) \leq e^{-2\epsilon^2 k}. \tag{4.44}$$

Combining (4.42) with (4.44) yields $P(\mathcal{X}_k^{(j)} > n_i) \leq e^{-2\epsilon^2 k}$ for $i \geq \max\{M(\delta), N(\delta)\}$ and $k \geq i$. Combining this bound with (4.41) yields (for large values of i and k):

$$\int_{\mathcal{X}_k^{(j)}>n_i} \mathcal{X}_k^{(j)}\, dP \leq a_0^{(j)} \sup_{k\geq i}\left\{\frac{e^{-2\epsilon^2 k}}{a_k}\right\}.$$

Since $e^{-2\epsilon^2 k}/a_k \to 0$ (by assumption), we know that $\sup_{k\geq i}\{e^{-2\epsilon^2 k}/a_k\} \to 0$ as $i \to \infty$. Therefore, the set $\{\mathcal{X}_k\}_{k\geq 0}$ is uniformly integrable and Theorem 4.5.4 in (Chung 2001) implies $E[\tilde{a}_k^{(j)}/a_k] \to \rho(j, q(j))$ as desired. □

The following corollary gives sufficient conditions for the iterates of Algorithm 2 to converge w.p.1 to θ^*, that is for $\hat{\theta}_k \to \theta^*$ w.p.1.

Corollary 2. Let $\hat{\theta}_k$ be defined by according to Algorithm 2 and assume the following conditions hold:

A0′ Let $a_k > 0$, $a_k^{(j)} > 0$, $\sum_{k=0}^{\infty} a_k = \infty$ and $\sum_{k=0}^{\infty} a_k^2 < \infty$. Additionally, assume the conditions of Lemmas 11 and 12 hold with $0 < \rho(j, q(j)) < \infty$.

A4′ For $k \geq 0$, let there exist a set $\Omega_1 \subset \Omega$ with $P(\Omega_1) = 1$ and a scalar $0 < R_1(\omega) < \infty$ such that the set $\{\hat{\theta}_k\}_{k \geq 0}$ is contained within a p-dimensional ball of radius $R_1(\omega)$ centered at the origin for all $\omega \in \Omega_1$.

A5′ For $k \geq 0$, let there exist a set $\Omega_2 \subset \Omega$ with $P(\Omega_2) = 1$ and a scalar $0 < R_2(\omega) < \infty$ such that the set $\{\beta_k^{(j_k)}(\hat{\theta}_k)\}_k$ is contained within a p-dimensional ball of radius $R_2(\omega)$ centered at the origin for all $\omega \in \Omega_2$. Furthermore, let $\beta_k^{(j_k)}(\hat{\theta}_k) \to 0$ w.p.1.

A6′ $\boldsymbol{D}_r \equiv \left(\frac{\tilde{a}_r^{(j_r)}}{a_r}\right) \xi_r^{(j_r)}(\hat{\theta}_r)$ and $\lim_{k \to \infty} P\left(\sup_{m \geq k} \left\| \sum_{r=k}^{m} a_r \boldsymbol{D}_r \right\| \geq \varepsilon\right) = 0$ for $\varepsilon > 0$.

A7′ Let θ^* be a locally asymptotically stable (in the sense of Lyapunov) solution of the differential equation:

$$\dot{\boldsymbol{Z}}(t) = -\sum_{j=1}^{d} q(j)\rho(j, q(j))\boldsymbol{g}^{(j)}(\boldsymbol{Z}(t)) = -\boldsymbol{\Lambda}\boldsymbol{g}(\boldsymbol{Z}(t)), \tag{4.45}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with ith diagonal entry $\Lambda_{ii} = q(j)\rho(j, q(j))$ where j is such that $i \in \mathcal{S}_j$. Let (4.45) have domain of attraction $DA(\theta^*)$.

A3′ and A8′ The same as A3 and A8, respectively.

Then, $\hat{\theta}_k \to \theta^*$ w.p.1.

Proof. We will show that conditions A0′ and A3′–A8′ imply A0–A8, the result will then follow from Theorem 2. First, note that A0′ along with the results from Lemmas 11 and 12 imply that A0 holds with $r_j = \rho(j, q(j))$. Additionally, condition A1 is satisfied because $s_k = 1$ and $n_k(m) = 1$. Next we show that condition A2 is satisfied. First, note that $x_k(j) = \chi\{j_k = j\}(\tilde{a}_k^{(j)}/a_k)$. Therefore, $\mu_k(j) = E[\chi\{j_k = j\}(\tilde{a}_k^{(j)}/a_k)]$. However, because $\chi\{j_k = j\}$ and $\tilde{a}_k^{(j)}$ are independent in Algorithm 2, $\mu_k(j) = q(j)E[\tilde{a}_k^{(j)}/a_k]$ so that $\lim_{k\to\infty} \mu_k(j) = q(j)\rho(j, q(j))$. Then, the first part of A2 holds with $\mu(j) = q(j)\rho(j, q(j))$. Next, for Algorithm 2 we have $S_r = \sum_{j=1}^{d} E[\tilde{a}_r^{(j)}/a_r](\chi\{j_r = j\} - q(j))$. Therefore, by the independence of the variables $\{j_k\}_{k\geq 0}$, we see that the rest of condition A2 holds. Since A3′–A8′ are simply versions of A3–A8 rewritten in terms of the notation of Algorithm 2, it follows that conditions A3–A8 are automatically satisfied. Therefore, the conclusion of Theorem 2 holds. □

Another special case of the GCSA algorithm pertains to the case where $\hat{\theta}_k$ is updated via a deterministic subvector update pattern. This is described in Algorithm 3. The basic idea is that the number of updates made during an iteration is a constant, s, and the mth block consists of $n(m)$ updates to subvector $j(m)$. In Algorithm 3 the variables s, $n(m)$, and $j(m)$ are deterministic (cyclic seesaw is a special case). The following corollary gives conditions for the convergence of the iterates of Algorithm 3.

Algorithm 3 Deterministic Pattern for Coordinate Selection

Require: $\hat{\theta}_0$, $\{a_k^{(j)}\}_{k\geq 0}$ for $j = 1, \ldots, d$, and let $\mathcal{S}_1, \ldots, \mathcal{S}_d$ be such that (4.36) holds. Set $k = 0$.

1: **while** stopping criterion has not been reached **do**

2: Let $s \in \mathbb{Z}^+$ be a finite integer-valued constant.

3: **for** $m = 1, \ldots, s$ **do**

4: Let $j(m) \in \{1, \ldots, d\}$ and let $n(m) \in \mathbb{Z}^+$ be a finite integer.

5: **for** $i = 1, \ldots, n(m)$ **do**

6: Define:

$$\hat{\theta}_k^{(I_{m,i})} \equiv \hat{\theta}_k - \sum_{z=1}^{m-1} \sum_{\ell=0}^{n(z)-1} \left[a_k^{(j(z))}\, \hat{\boldsymbol{g}}^{(j(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) \right] - \sum_{\ell=0}^{i-1} \left[a_k^{(j(m))}\, \hat{\boldsymbol{g}}^{(j(m))} \left(\hat{\theta}_k^{(I_{m,\ell})} \right) \right]$$

7: **end for**

8: **end for**

9: Let $\hat{\theta}_{k+1} \equiv \hat{\theta}_k^{(I_{m,i})}$ with $m = s$ and $i = n(s)$.

10: set $k = k + 1$

11: **end while**

Corollary 3. Let $\hat{\theta}_k$ be defined as in Algorithm 3. Assume the following hold:

A0″ Let $a_k > 0$, $a_k^{(j)} > 0$, $\sum_{k=0}^{\infty} a_k = \infty$ and $\sum_{k=0}^{\infty} a_k^2 < \infty$. Additionally, assume $a_k^{(j)}/a_k \to r_j$ with $0 < r_j < \infty$.

A4″ For $k \geq 0$, $m \leq s$ (s is defined in Line 2 of Algorithm 3), $i \leq n(m)$, let there exist a set $\Omega_1 \in \Omega$ with $P(\Omega_1) = 1$ and a scalar $0 < R_1(\omega) < \infty$ such that the set $\{\hat{\theta}_k^{(I_{m,i})}\}_{k,m,i}$ is contained within a p-dimensional ball of radius $R_1(\omega)$ centered at the origin for all $\omega \in \Omega_1$.

A5″ For $k \geq 0$, $m \leq s$, $i \leq n(m)$, let there exist a set $\Omega_2 \in \Omega$ with $P(\Omega_2) = 1$ and a scalar $0 < R_2(\omega) < \infty$ such that the set $\{\beta_k^{(j(m))}(\hat{\theta}_k^{(I_{m,i})})\}_{k,m,i}$ is contained within a p-dimensional ball of radius $R_2(\omega)$ centered at the origin for all $\omega \in \Omega_2$. Furthermore, let $\beta_k^{(j(m))}(\hat{\theta}_k^{(I_{m,i})}) \to 0$ w.p.1 uniformly in m and i.

A6″ Define $\boldsymbol{D}_r \equiv \sum_{m=1}^{s} \sum_{i=0}^{n(m)-1} \left(\frac{a_r^{(j(m))}}{a_r}\right) \xi_r^{(j(m))} \left(\hat{\theta}_r^{(I_{m,i})}\right)$. Assume for all $\varepsilon > 0$ we have $\lim_{k\to\infty} P\left(\sup_{m\geq k} \left\| \sum_{r=k}^{m} a_r \boldsymbol{D}_r \right\| \geq \varepsilon\right) = 0$.

A7″ θ^* is a locally asymptotically stable (in the sense of Lyapunov) solution of:

$$\dot{\boldsymbol{Z}}(t) = -\sum_{j=1}^{d} \left[r_j \sum_{m=1}^{s} \chi\{j(m) = j\} n(m) \right] \boldsymbol{g}^{(j)}(\boldsymbol{Z}(t)) = -\boldsymbol{\Lambda} \boldsymbol{g}(\boldsymbol{Z}(t)), \qquad (4.46)$$

where the ith diagonal entry of $\boldsymbol{\Lambda}$ equals $\Lambda_{ii} = r_j \sum_{m=1}^{s} \chi\{j(m) = j\} n(m)$ where j is such that $i \in \mathcal{S}_j$. Let (4.46) have domain of attraction $DA(\theta^*)$.

A3″ and A8″ The same as A3 and A8, respectively.

Then, $\hat{\theta}_k \to \theta^*$ w.p.1.

Proof. Because all gain sequences are deterministic (not random), condition A0″ implies A0. Additionally, since $s_k = s < \infty$ and since $n_k(m) = n(m)$ is a bounded deterministic function of m, then condition A1 is also satisfied. Next, note that for Algorithm 3:

$$\mu_k(j) = \left(\frac{a_k^{(j)}}{a_k}\right) \sum_{m=1}^{s} \chi\{j(m) = j\} n(m).$$

Here, $\mu_k(j)$ is a deterministic quantity since the subvector to update (and hence the indicator random variables in the previous equation) are deterministic functions of m. Since $a_k^{(j)}/a_k \to r_j$ then $\mu(j) = r_j \sum_{m=1}^{s} \chi\{j(m) = j\} n(m)$. Thus, the first part of A2 holds. The second part of A2 holds since $S_r = 0$ for all r. Because A3″–A8″ are versions of A3–A8 that have been rewritten in terms of Algorithm 3, we have shown conditions A0–A8 are satisfied. □

The following section discusses the validity of conditions A0–A8.

4.5 On the Convergence Conditions

Sections 4.2–4.4 presented conditions for the convergence of the GCSA algorithm (conditions A0–A8). It is worthwhile to note that conditions A0–A8 closely resemble those in Kushner and Clark (1978) for the convergence of SA procedures. This is not surprising given that the proof of Theorem 2 is based on principles similar to those in the proof of Theorem 2.3.1 in Kusher and Clark (1978) . This section comments on the validity of A0–A8.

Condition A0. Requiring $a_k > 0$ and $a_k^{(j)} > 0$ is a reasonable assumption when one has complete control over the specific form of the gain sequences (in the deterministic steepest descent algorithm, having the gain sequences be strictly positive guarantee that the update direction is in fact a descent direction). Furthermore, the requirement that $\sum_{k=0}^{\infty} a_k = \infty$ and $\sum a_k^2 < \infty$ is

easily satisfied by sequences of the form $a_k = a/(1+k+A)^\alpha$ where $a > 0$ and $A > 0$ and where $0.5 < \alpha \leq 1$ (the non-convergence and square summability for series associated with a_k is a long-standing requirement in SA).

Another assumptions in condition A0 is that the sequence $\tilde{a}_k^{(j)}/a_k$ converges in expectation and w.p.1 of to some finite, strictly positive constant. Because $\tilde{a}_k^{(j)}/a_k$ must converge (in a stochastic sense) to a non-zero constant for all j, all gain sequences must converge to zero at the same rate (in a stochastic sense). For Algorithm 2 (randomized subvector selection), Lemmas 11 and 12 give conditions under which $\tilde{a}_j^{(j)}/a_k$ converges w.p.1 and in expectation to a finite constant. For Algorithm 3, it suffices to have $a_k^{(j)} = a^{(j)}/(k + A^{(j)} + 1)^\alpha$ and $a_k = a/(k+1)^\alpha$, where $a^{(j)} > 0$, $a > 0$, $A^{(j)} > 0$, and $0 < \alpha \leq 1$ (α must be the same for all j). It is important to note that the sequence a_k is not used by the GCSA algorithm. Rather, a_k serves only as a representation of the rate at which $a_k^{(j)}$ decreases. It is important to note that the requirement in condition A0 that r_j be strictly positive is introduced only to guarantee that $\sum_{j=1}^d \mu(j)\boldsymbol{g}^{(j)}(\theta) = \mathbf{0}$ only when $\boldsymbol{g}(\theta) = \mathbf{0}$. This implies that the only way in which the ODE in (4.15) is equal to zero is if $\boldsymbol{Z}(t) = \theta^*$. Note that if the sets $\mathcal{S}_1, \ldots, \mathcal{S}_d$ overlap, it is still possible that the only solution to $\sum_{j=1}^d \mu(j)\boldsymbol{g}^{(j)}(\theta) = \mathbf{0}$ occurs at $\theta = \theta^*$ even when some of the $\mu(j)$ are equal to zero (this occurs provided $\sum_{j=1}^d \mu(j)\boldsymbol{g}^{(j)}(\theta) = \boldsymbol{\Lambda}\boldsymbol{g}(\theta)$ for some positive definite diagonal matrix $\boldsymbol{\Lambda}$).

Conditions A1 and A2. Condition A1 requires that the number of blocks

within the k iteration, s_k, and the number of updates within each block, $n_k(m)$, be uniformly bounded over k, m, and ω. A special case where A1 holds is when a hard bound is imposed on s_k and $n_k(m)$, as in the case of Algorithms 2 and 3. Additionally, both these algorithms were shown to satisfy condition A2, which imposes a restriction on how s_j and $n_j(m)$ relate to s_i and $n_i(m)$ for $i \neq j$.

Conditions A3 and A4. Despite the fact that the gradient of $L(\theta)$ is not explicitly used by the GCSA algorithm, the convergence theory of Section 4.3 requires the existence of $\boldsymbol{g}(\theta)$ and its continuity, a fact which may be hard to verify. Condition A4 is also difficult to verify in practice. Here, it is assumed that the algorithm's iterates are bounded w.p.1 (see Figure 4.3). While the boundedness of the iterates is a common assumption throughout the SA, it remains somewhat controversial (e.g., Benveniste et al. 1990, p. 46). Kushner and Clark (1978), however, mention that this boundedness is not necessarily a strong assumption since one typically imposes bounds on θ in practice. Projected versions of GCSA could be the subject of future work and, with this in mind, the following quote seems fitting:

> There is no general scheme for showing $P(Q) = 1$ [the iterates of a realization are bounded w.p.1]. There are, however, problem-specific techniques for special problem classes. . . . One way to escape the boundedness issue is to alter the algorithm by projecting the iterates back onto a prescribed, large bounded set whenever they exit from the same. The trade-off is that the limiting ODE becomes more complicated. It is now confined to the said set and thus involves a "reflection at the boundary" of the same in an appropriate sense. (Borkar 1998, on an asynchronous SA algorithm related to GCSA).

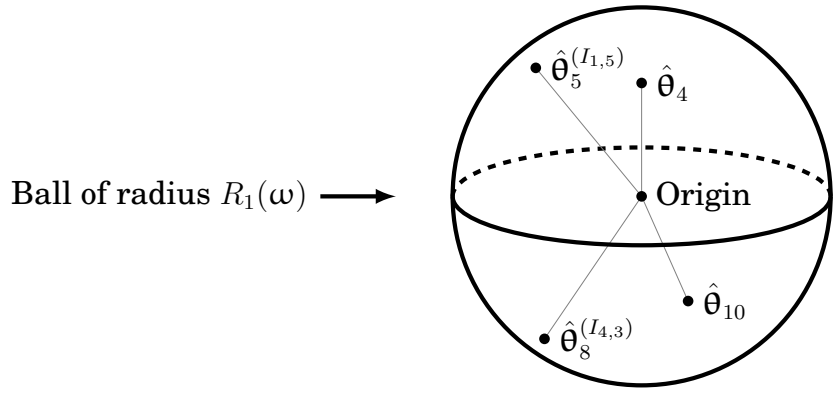


Figure 4.3: Condition A4 states that the iterates produced by a single realization of GCSA must be contained within a ball of radius $R_1(\omega) < \infty$ for $\omega \in \Omega_1$

Condition A5. When the GCSA update directions are obtained using SG-type measurements, all bias terms are identically zero and condition A5 would automatically be satisfied. For the cyclic seesaw SPSA algorithm, Spall (1992) gives conditions under which the bias term is bounded (w.p.1) and converges to zero. In a manner analogous to Lemma 1 in Spall (1992) and under similar conditions (after a natural adaptation for the cyclic setting) it can be shown that the bias terms of the GCSA algorithm satisfy A5.

Condition A6. This condition restricts the relationship between a_k and $\boldsymbol{D}_k$, a vector which represents the "total noise" of the $(k+1)$st iteration of GCSA (see the statement of condition A6 for a the precise definition of $\boldsymbol{D}_k$). Let us give an example of when condition A6 holds. Assume that for each $k \geq 0$ the sequence $\{\sum_{r=k}^{m} a_r \boldsymbol{D}_r\}_{m \geq k}$ (indexed by m) is a martingale sequence, that $E\|\boldsymbol{D}_k\|^2 < \infty$, and that $E[\boldsymbol{D}_i^\top \boldsymbol{D}_j] = 0$ for $i \neq j$. Then, by a relation on p. 315 in Doob (1953)

we obtain the following inequality:

$$\lim_{k\to\infty} P\left(\sup_{m\geq k}\left\|\sum_{r=k}^{m} a_r \boldsymbol{D}_r\right\| \geq \varepsilon\right) \leq \lim_{k\to\infty} \varepsilon^{-2}\sum_{r=k}^{\infty} a_r^2 E\|\boldsymbol{D}_r\|^2. \tag{4.47}$$

If in addition we assume $\sum_{k=1}^{\infty} a_k^2 < \infty$ and $E\|\boldsymbol{D}_k\|^2 < \sigma^2$ for some finite $\sigma^2 > 0$, then the limit of the upper bound in (4.47) is equal to zero and A6 holds. Requiring the noise term to have a bounded variance could be a reasonable assumption when the update directions are obtained using SG-based gradient measurements. However, it is not always reasonable to assume the noise has bounded variance. For example, for the SPSA algorithm (which is a special case of GCSA) it is often the case that the variance of $\boldsymbol{D}_k$ grows at a rate proportional to $1/c_k^2$ (recall $c_k \to 0$). Therefore, in order for the limit in the upper bound of (4.47) to be zero, it would be necessary to have a_k and c_k satisfy $\sum_{k=1}^{\infty} a_k^2/c_k^2 < \infty$ (see, for example, Spall 2003, p. 183).

Conditions A7 and A8. Since the premise of stochastic optimization is that the function $L(\theta)$ and its gradient are unknown, conditions A7 and A8 are likely to be impossible to verify in practice. Nevertheless, it is impractical to attempt to derive convergence conditions for an optimization algorithm without imposing restrictions on the properties of the function to minimize. Informally, conditions A7 and A8 force $\hat{\theta}_k$ be inside the domain of attraction of θ^* with enough frequency (for infinitely many k) that the iterates begin to bene-

fit from the asymptotic stability of θ^*, which results in $\hat{\theta}_k \to \theta^*$ w.p.1. A few observations regarding the ODE in condition A7 are given next.

Understanding the ODE in (4.15) requires understanding the terms $\mu(j)$ (defined on p. 52). When $\tilde{a}_k^{(j)}/a_k$ is independent of s_k, $n_k(m)$, and $j_k(m)$ (a valid assumption for Algorithms 2 and 3), $\mu(j)$ can be rewritten as:

$$\mu(j) = r_j \times [\text{asymptotic average \# times entries } \mathcal{S}_j \text{ are updated per-iteration}].$$

By construction, $\mu(j) > 0$ for all j. Thus, there exists a diagonal matrix Λ with strictly positive diagonal entries (determined by the $\mu(j)$) such that:

$$\dot{\boldsymbol{Z}}(t) = -\Lambda \boldsymbol{g}(\boldsymbol{Z}(t)) \tag{4.48}$$

(see p. 84 and p. 87 for two special cases of Λ). The ODE in (4.48) is a slight variation of the ODE that arises in the standard SA algorithm from (2.3), where $\Lambda = \boldsymbol{I}$ (see, for example, Spall 2003, Chapter 7).

At this point one might wonder whether θ^*, a minimizer of $L(\theta)$, is necessarily a stable solution to (4.48). The answer, unfortunately, is no. While $\boldsymbol{Z}(t) = \theta^*$ is certainly an *equilibrium* point of (4.48) (and hence a solution to (4.48)), it is entirely possible for $\boldsymbol{Z}(t) = \theta^*$ not to be a *stable* equilibrium point of (4.48) even when $L(\theta)$ is differentiable for all degrees of differentiation) and $\Lambda = \boldsymbol{I}$ (e.g., Absil and Kurdyka 2006, Proposition 2). Thus, condition A7 is not

automatically satisfied when θ^* is a minimizer of $L(\theta)$. We will note, however, that if θ^* is a Lyapunov stable solution to $\dot{\boldsymbol{Z}}(t) = -\boldsymbol{g}(\boldsymbol{Z}(t))$, if the entries of $\boldsymbol{\Lambda}$ have finite magnitudes, and if the smallest eigenvalue of $\boldsymbol{\Lambda}$ is strictly positive (more generally, if the smallest real part of an eigenvalue is strictly positive), then θ^* is also a Lyapunov stable solution to (4.48) (e.g., Benveniste et al. 1990, pp. 111–112). Therefore, asking for θ^* to be a stable solution of (4.48) is not a stronger condition than the corresponding stability assumption when $\boldsymbol{\Lambda} = \boldsymbol{I}$.

In general, it is impossible to know whether there is a unique solution to $\boldsymbol{g}(\theta) = 0$. However, if $\theta^{*,1}$ and $\theta^{*,2}$ denote two distinct zeros of the gradient, it is impossible for both vectors to satisfy A7 and A8. Therefore, the GCSA iterates will converge to whichever of $\theta^{*,1}$ and $\theta^{*,2}$ (if any) satisfies both A7 and A8. Based on the idea in Kushner and Clark (1978, p.39), however, condition A8 could be relaxed to allow all zeros of the gradient to be contained within some set that is stable (in the sense of Lyapunov), which would result in the GCSA algorithm's iterates converging to the aforementioned set w.p.1. This, however, would not guarantee convergence to a zero of $\boldsymbol{g}(\theta)$.

4.6 Concluding Remarks

This chapter gave a theorem for the convergence of the GCSA algorithm (Theorem 2) as well as two corollaries pertaining to special cases of GCSA

(Corollaries 2 and 3). The proof of Theorem 2 is based on the proof of Theorem 2.3.1 in Kushner and Clark (1978) which, in turn, is based on the ODE-based method for proving convergence of SA algorithms that was introduced by Ljung (1977). The time-dependent recursion in (4.14), which describes the evolution of the GCSA iterates, is similar to equation (2.3.3) in Kushner and Clark (1978) although we point out that (4.14) is significantly more complex. As a result, some of the conditions of Theorem 2 resemble the conditions of Kushner and Clark's Theorem 2.3.1.

In contrast to this chapter's focus on convergence w.p.1 of the GCSA iterates, the following chapter focuses on the asymptotic normality of the normalized iterates from Algorithm 3, a special case of GCSA in which the subvector to update is selected according to a deterministic pattern (see, for example, the algorithm specified by (3.15a–f), which is a special case of Algorithm 3).

Chapter 5

Asymptotic Normality of GCSA

When applicable, asymptotic normality results can be used to construct approximate confidence regions for the SA iterates, to compute the relative efficiency between two SA algorithms by comparing their asymptotic mean squared errors (e.g., Spall 1992), or even to define the rate of convergence of the vector-valued random sequence $\hat{\theta}_k$. Fabian (1968), a seminal paper in the SA literature, provides conditions for the asymptotic normality of the iterates of SA algorithms (after an appropriate centering and scaling). Fabian's theorem (Fabian 1968, Theorem 2.2), gives conditions under which there exists a constant $\beta > 0$ such that $k^{\beta/2}(\hat{\theta}_k - \theta^*)$ has a limiting multivariate normal distribution with a theoretically computable mean and covariance. One of the assumptions from Fabian's theorem, however, is too strong for the GCSA algorithm. Specifically, Fabian (1968, Assumption 2.2.1) requires a matrix that af-

fects the mean and covariance matrix in the limiting distribution of $k^{\beta/2}(\hat{\theta}_k - \theta^*)$ to be symmetric. Section 5.3 generalizes Fabian's result by relaxing the symmetry assumption of the aforementioned matrix. The resulting generalization expands the theorem's applicability to include a broader range of SA algorithms of practical interest (see Appendix A) including a special case of GCSA.

The remainder of this chapter is organized as follows. First, Section 5.1 reviews Fabian's theorem and its connection to SA algorithms. Then, Section 5.2 explains the nature of the incompatibility between Fabian's theorem and the GCSA algorithm. Section 5.3 provides a generalization of Fabian's theorem which is used in Section 5.4 to derive an asymptotic normality result for a special case of GCSA. Section 5.5 comments on the validity of the assumptions in Section 5.4. Lastly, Section 5.6 contains concluding remarks.

5.1 Reviewing Fabian's Theorem

This section briefly reviews Fabian's result on asymptotic normality.

Theorem 3 (Fabian 1968, Theorem 2.2 or "Fabian's Theorem")**.** For $k \geq 1$, let $\boldsymbol{V}_k$, $\boldsymbol{W}_k$, $\boldsymbol{T}_k$, and $\boldsymbol{T}$ be vectors in $\mathbb{R}^p$, and let $\boldsymbol{\Gamma}_k$, $\boldsymbol{\Phi}_k$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Phi}$, and $\boldsymbol{P}$ be matrices in $\mathbb{R}^{p\times p}$. Let $\boldsymbol{W}_k$ satisfy the recursion:

$$\boldsymbol{W}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\boldsymbol{\Gamma}_k)\boldsymbol{W}_k + \frac{\boldsymbol{T}_k}{k^{\alpha+\beta/2}} + \frac{\boldsymbol{\Phi}_k \boldsymbol{V}_k}{k^{(\alpha+\beta)/2}}. \tag{5.1}$$

Additionally, let $\mathcal{F}_k$ be a non-decreasing sequence of σ-fields and assume there exists a set $\mathcal{S}$ such that $\mathcal{F}_k \subset \mathcal{S}$ for all k. Assume the following conditions hold:

B0 $\boldsymbol{\Gamma}_k$, $\boldsymbol{\Phi}_{k-1}$, and $\boldsymbol{V}_{k-1}$ are $\mathcal{F}_k$-measurable.

B1 $\boldsymbol{\Gamma}_k \to \boldsymbol{\Gamma}$ w.p.1 where $\boldsymbol{\Gamma} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top$ for some real orthogonal matrix $\boldsymbol{P}$ and a diagonal matrix $\boldsymbol{\Lambda}$ with strictly positive eigenvalues (an important implication is that the matrix $\boldsymbol{\Gamma}$ must be symmetric).

B2 $\boldsymbol{\Phi}_k \to \boldsymbol{\Phi}$ w.p.1.

B3 Either $\boldsymbol{T}_k \to \boldsymbol{T}$ w.p.1 or $E\|\boldsymbol{T}_k - \boldsymbol{T}\| \to 0$.

B4 $E[\boldsymbol{V}_k|\mathcal{F}_k] = \boldsymbol{0}$ and there exists C such that $C > \|E[\boldsymbol{V}_k\boldsymbol{V}_k^\top|\mathcal{F}_k] - \boldsymbol{\Sigma}\| \to 0$.

B5 $\sigma_{k,r}^2 \equiv E\chi\{\|\boldsymbol{V}_k\|^2 \geq rk^\alpha\}\|\boldsymbol{V}_k\|^2$ (recall $\chi\{\mathcal{E}\}$ denotes the indicator function of the event $\mathcal{E}$). For every $r > 0$ assume that one of the following holds:

(i) $\lim_{k\to\infty} \sigma_{k,r}^2 = 0$.

(ii) $\alpha = 1$ and $\lim_{n\to\infty} n^{-1} \sum_{k=1}^n \sigma_{k,r}^2 = 0$.

B6 Let $\lambda \equiv \min_i \{\Lambda_{ii}\}$ where Λ_{ii} is the ith diagonal entry of $\boldsymbol{\Lambda}$. Let α and β be constants such that $0 < \alpha \leq 1$ and $0 \leq \beta$. Define $\beta_+ \equiv \beta$ if $\alpha = 1$ and $\beta_+ \equiv 0$ otherwise. Let $\beta_+ < 2\lambda$.

Then, the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ is a multivariate normal random variable with mean $(\boldsymbol{\Gamma} - (\beta_+/2)\boldsymbol{I})^{-1}\boldsymbol{T}$ and covariance matrix $\boldsymbol{P}\boldsymbol{Q}\boldsymbol{P}^\top$, where the

(i,j)th entry of $\boldsymbol{Q}$ is equal to $(\boldsymbol{P}^\top \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top \boldsymbol{P})_{ij}(\Lambda_{ii}+\Lambda_{jj}-\beta_+)^{-1}$ with $(\boldsymbol{P}^\top \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top \boldsymbol{P})_{ij}$ denoting the (i,j)th entry of the matrix $\boldsymbol{P}^\top \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top \boldsymbol{P}$.

Let us discuss the connection between Fabian's theorem (Theorem 3) and stochastic optimization. Consider a stochastic optimization algorithm that produces updates according to (2.4). Writing $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$ as in (2.5) and letting $\boldsymbol{W}_k = \hat{\theta}_k - \theta^*$, the algorithm in (2.4) can be rewritten as follows:

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k - a_k \left(\boldsymbol{g}(\hat{\theta}_k) + \boldsymbol{\beta}_k(\hat{\theta}_k) + \boldsymbol{\xi}_k(\hat{\theta}_k) \right). \tag{5.2}$$

Asssume $L(\theta)$ is twice continuously differentiable and denote its Hessian matrix by $\boldsymbol{H}(\theta)$. Then, using Taylor's theorem we may write $\boldsymbol{g}(\hat{\theta}_k) = \tilde{\boldsymbol{H}}_k(\hat{\theta}_k - \theta^*)$, where the ith row of $\tilde{\boldsymbol{H}}_k$ is equal to the ith row of $\boldsymbol{H}(\theta)$ evaluated at $\theta = (1-\lambda_i)\hat{\theta}_k + \lambda_i \theta^*$ for some $\lambda_i \in [0,1]$ which depends on $\hat{\theta}_k$. By expanding $\boldsymbol{g}(\hat{\theta}_k)$ around θ^* in this way, we can rewrite (5.2) in the following manner:

$$\boldsymbol{W}_{k+1} = \left(\boldsymbol{I} - k^{-\alpha} k^{\alpha} a_k \tilde{\boldsymbol{H}}_k \right) \boldsymbol{W}_k - \frac{k^{\alpha+\beta/2} a_k \boldsymbol{\beta}_k(\hat{\theta}_k)}{k^{\alpha+\beta/2}} - \frac{k^{(\alpha+\beta)/2} a_k \boldsymbol{\xi}_k(\hat{\theta}_k)}{k^{(\alpha+\beta)/2}}. \tag{5.3}$$

Letting $\boldsymbol{\Gamma}_k = k^{\alpha} a_k \tilde{\boldsymbol{H}}_k$, $\boldsymbol{\Phi}_k = \boldsymbol{I}$, $\boldsymbol{T}_k = -k^{\alpha+\beta/2} a_k \boldsymbol{\beta}_k(\hat{\theta}_k)$, and $\boldsymbol{V}_k = -k^{(\alpha+\beta)/2} a_k \boldsymbol{\xi}_k(\hat{\theta}_k)$, it follows that (5.3), and therefore (2.4), can be rewritten in the form of (5.1). Consequently, Fabian's theorem can be used to derive conditions for the asymptotic normality of $k^{\beta/2}(\hat{\theta}_k - \theta^*)$ when $\hat{\theta}_k$ is obtained via the general SA recursion in (2.4).

Undoubtedly, Fabian's theorem is already applicable to a variety of SA algorithms (see, for example, Zhou and Hu 2014, Kar et al. 2013, Hu et al. 2012, and Zorin et al. 2000 to name a few recent applications). However, a critical assumption in Fabian's theorem is that $\boldsymbol{\Gamma}_k \to \boldsymbol{\Gamma}$ w.p.1 for some real, positive definite matrix $\boldsymbol{\Gamma}$ (condition B1). This assumption gives rise to an important complication when attempting to write the GCSA algorithm in the form of (5.1); the nature of this complication is explored in the following section.

5.2 A Limitation of Fabian's Theorem

In this section we explore an issue that arises when attempting to write the recursion for the GCSA algorithm (see Line 7 of Algorithm 1) in the form of (5.1), the recursion in Fabian's theorem. The nature of this issue is simple: the matrix $\boldsymbol{\Gamma}$ from condition B1 cannot generally be assumed to be symmetric for the GCSA algorithm, thus assumption B1 is not always satisfied and Fabian's theorem is not applicable. Using a simple example, we first explain why $\boldsymbol{\Gamma}$ cannot be assumed to be symmetric for the GCSA algorithm. Then, we discuss why it is not always possible to redefine $\boldsymbol{\Gamma}_k$, $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{V}_k$ in such a way that $\boldsymbol{\Gamma}$ (the limit w.p.1 of $\boldsymbol{\Gamma}_k$) is symmetric. Finally, we propose a slight generalization of Fabian's theorem that would allow for treatment of the special case of the GCSA algorithm where the subvector to update is selected according to a

deterministic pattern (see, for example, the algorithm defined by (3.17)).

Let us begin by rewriting the cyclic seesaw SA algorithm (a special case of GCSA) in the form of (5.1). Recall that cyclic seesaw SA satisfies:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k^{(1)} \hat{\boldsymbol{g}}_k^{(1)}(\hat{\theta}_k) - a_k^{(2)} \hat{\boldsymbol{g}}_k^{(2)}(\hat{\theta}_k^{(I)}), \tag{5.4}$$

where $\hat{\theta}_k^{(I)}$ is defined in (3.3). If $L(\theta)$ is twice continuously differentiable then expanding $\boldsymbol{g}(\hat{\theta}_k)$ around θ^* and $\boldsymbol{g}(\hat{\theta}_k^{(I)})$ around $\hat{\theta}_k$:

$$\begin{aligned}
\boldsymbol{g}^{(1)}(\hat{\theta}_k) &= \boldsymbol{J}^{(1)}(\overline{\theta}_k)(\hat{\theta}_k - \theta^*), \\
\boldsymbol{g}^{(2)}(\hat{\theta}_k) &= \boldsymbol{J}^{(2)}(\overline{\theta}_k)(\hat{\theta}_k - \theta^*), \\
\boldsymbol{g}^{(2)}(\hat{\theta}_k^{(I)}) &= \boldsymbol{g}^{(2)}(\hat{\theta}_k) + \boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})(\hat{\theta}_k^{(I)} - \hat{\theta}_k),
\end{aligned}$$

where $\boldsymbol{J}^{(j)}(\theta)$ is the Jacobian of $\boldsymbol{g}^{(j)}(\theta)$ with respect to θ, the ith row of $\boldsymbol{J}^{(j)}(\overline{\theta}_k)$ is equal to the ith row of $\boldsymbol{J}^{(j)}(\theta)$ evaluated at $\theta = (1-\lambda_i)\hat{\theta}_k + \lambda_i \theta^*$ for some $\lambda_i \in [0,1]$ which depends on $\hat{\theta}_k$, and the ith row of $\boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})$ is the ith row of $\boldsymbol{J}^{(2)}(\theta)$ with $\theta = (1-\lambda_i)\hat{\theta}_k^{(I)} + \lambda_i \hat{\theta}_k$ for some $\lambda_i \in [0,1]$ which depends on $\hat{\theta}_k^{(I)}$. We now rewrite (5.4) as:

$$\begin{aligned}
\hat{\theta}_{k+1} = \hat{\theta}_k &- a_k^{(1)} \boldsymbol{J}^{(1)}(\overline{\theta}_k)(\hat{\theta}_k - \theta^*) - a_k^{(2)} \boldsymbol{J}^{(2)}(\overline{\theta}_k)(\hat{\theta}_k - \theta^*) - a_k^{(2)} \boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})(\hat{\theta}_k^{(I)} - \hat{\theta}_k) \\
&- a_k^{(1)} \left(\beta_k^{(1)}(\hat{\theta}_k) + \xi_k^{(1)}(\hat{\theta}_k) \right) - a_k^{(2)} \left(\beta_k^{(2)}(\hat{\theta}_k^{(I)}) + \xi_k^{(2)}(\hat{\theta}_k^{(I)}) \right).
\end{aligned} \tag{5.5}$$

Using (3.3) it follows that $-(\hat{\theta}_k^{(I)} - \hat{\theta}_k)/a_k^{(1)} = \boldsymbol{g}^{(1)}(\hat{\theta}_k) + \boldsymbol{\beta}_k^{(1)}(\hat{\theta}_k) + \boldsymbol{\xi}_k^{(1)}(\hat{\theta}_k)$. Combining this last observation with (5.5) and using (2.5), the cyclic seesaw SA algorithm can be written in the form of (5.1) by letting $\boldsymbol{W}_k = \hat{\theta}_k - \theta^*$,

$$\boldsymbol{\Gamma}_k = k^{\alpha}\left[a_k^{(1)}\boldsymbol{J}^{(1)}(\overline{\theta}_k) + a_k^{(2)}\boldsymbol{J}^{(2)}(\overline{\theta}_k)\right], \tag{5.6a}$$

$$\begin{aligned}\boldsymbol{T}_k = &- k^{\alpha+\beta/2}\Big[a_k^{(1)}\boldsymbol{\beta}_k^{(1)}(\hat{\theta}_k) + a_k^{(2)}\boldsymbol{\beta}_k^{(2)}(\hat{\theta}_k^{(I)}) - a_k^{(1)}a_k^{(2)}\boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})\boldsymbol{g}^{(1)}(\hat{\theta}_k)\\ &- a_k^{(1)}a_k^{(2)}\boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})\boldsymbol{\beta}_k^{(1)}(\hat{\theta}_k) - a_k^{(1)}a_k^{(2)}E\Big(\boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})\boldsymbol{\xi}_k^{(1)}(\hat{\theta}_k)\Big|\mathcal{F}_k\Big)\Big],\end{aligned} \tag{5.6b}$$

$$\begin{aligned}\boldsymbol{V}_k = &- k^{(\alpha+\beta)/2}\Big[a_k^{(1)}\boldsymbol{\xi}_k^{(1)}(\hat{\theta}_k) + a_k^{(2)}\boldsymbol{\xi}_k^{(2)}(\hat{\theta}_k^{(I)}) + a_k^{(1)}a_k^{(2)}E\Big(\boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})\boldsymbol{\xi}_k^{(1)}(\hat{\theta}_k)\Big|\mathcal{F}_k\Big)\\ &- a_k^{(1)}a_k^{(2)}\boldsymbol{J}^{(2)}(\overline{\theta}_k^{(I)})\boldsymbol{\xi}_k^{(1)}(\hat{\theta}_k)\Big],\end{aligned} \tag{5.6c}$$

where $\mathcal{F}_k \equiv \{\hat{\theta}_0, \hat{\theta}_0^{(I)}, \hat{\theta}_1, \hat{\theta}_1^{(I)}, \ldots, \hat{\theta}_k\}$, and letting $\boldsymbol{\Phi}_k = \boldsymbol{I}$.

In general, the matrix $\boldsymbol{\Gamma}_k$ in (5.6a) cannot be assumed to converge to a symmetric matrix w.p.1. To see this consider the case where $k^{\alpha}a_k^{(1)} \to a^{(1)}$, $k^{\alpha}a_k^{(2)} \to a^{(2)}$, and $\boldsymbol{J}^{(j)}(\hat{\theta}_k) \to \boldsymbol{J}^{(j)}(\theta^*)$ w.p.1 (e.g., when $L(\theta)$ is twice continuously differentiable and $\hat{\theta}_k \to \theta^*$ w.p.1). Then,

$$\boldsymbol{\Gamma}_k \to \boldsymbol{\Gamma} = a^{(1)}\boldsymbol{J}^{(1)}(\theta^*) + a^{(2)}\boldsymbol{J}^{(2)}(\theta^*) \text{ w.p.1.} \tag{5.7}$$

If $a^{(1)} = a^{(2)} = a > 0$ then $\boldsymbol{\Gamma} = a\boldsymbol{H}(\theta^*)$ is clearly a symmetric matrix. This is not generally the case, however, when $a^{(1)} \neq a^{(2)}$ (an exception being the case where $L(\theta)$ is linearly separable in $\theta^{(1)}$ and $\theta^{(2)}$). Because there is no *unique* way to

define the variables Γ_k, $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{V}_k$ in (5.1), it may be tempting to think that it is always possible to redefine these variables so that Γ is symmetric. Next we show that such a redefinition often leads to very strong assumptions on $\hat{\theta}_k$.

Suppose an SA algorithm can be written in the form of (5.1) and that all of the conditions of Fabian's theorem are satisfied *with the exception that* Γ *is not symmetric*. For simplicity, let us consider a special case where writing the algorithm in the form of (5.1) can be done by letting $\beta \neq 0$, $\boldsymbol{T}_k = \boldsymbol{T}$, $\boldsymbol{\Phi}_k = \boldsymbol{I}$, and $\Gamma_k = \Gamma$ where Γ is not symmetric (note that we are assuming that $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and Γ_k do not depend on k). Now, assume the matrices Γ'_k, $\boldsymbol{\Phi}'_k$ and vectors $\boldsymbol{T}'_k$, $\boldsymbol{V}'_k$ provide an alternative way to write the SA algorithm in the form of (5.1). Furthermore, assume that Γ'_k converges w.p.1 to a positive definite matrix. Then, since

$$\boldsymbol{W}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\Gamma'_k)\boldsymbol{W}_k + \frac{\boldsymbol{T}}{k^{\alpha+\beta/2}} + \frac{\boldsymbol{V}_k}{k^{(\alpha+\beta)/2}} + k^{-\alpha}(\Gamma'_k - \Gamma)\boldsymbol{W}_k,$$

it is known that either $\boldsymbol{T}'_k$ must depend on $(\Gamma'_k - \Gamma)\boldsymbol{W}_k$ or $\boldsymbol{\Phi}'_k\boldsymbol{V}'_k$ must depend on $(\Gamma'_k - \Gamma)\boldsymbol{W}_k$. However, the assumptions Fabian's theorem imposes on $\boldsymbol{T}'_k$, $\boldsymbol{\Phi}'_k$, and $\boldsymbol{V}'_k$ are typically incompatible with the term $k^{\beta/2}(\Gamma'_k - \Gamma)\boldsymbol{W}_k$. Say, for example, that we let $\boldsymbol{\Phi}'_k = \boldsymbol{I}$, $\boldsymbol{V}'_k = \boldsymbol{V}_k$, and $\boldsymbol{T}'_k = \boldsymbol{T} + k^{\beta/2}(\Gamma'_k - \Gamma)\boldsymbol{W}_k$. Here, having $\boldsymbol{T}'_k$ converge to some finite vector $\boldsymbol{T}'$ w.p.1 (as required by B3) would impose a priori conditions on the stochastic rate at which $\hat{\theta}_k$ converges to θ^*. However, such a

condition violates the very purpose of Fabian's theorem, which is to establish such a rate of convergence. Alternatively, having $\boldsymbol{\Phi}'_k \boldsymbol{V}'_k = \boldsymbol{V}_k + k^{(\beta-\alpha)/2}(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k$ does not lead to an appropriate definition of $\boldsymbol{\Phi}'_k$ and $\boldsymbol{V}'_k$ given the restriction that $\boldsymbol{V}'_k$ must have mean zero conditionally on $\mathcal{F}_k$. By simply generalizing the theorem to relax the symmetry condition on $\boldsymbol{\Gamma}$ we avoid the need for imposing additional restrictions on $\hat{\theta}_k$. Next we propose a generalization of Fabian's theorem that will allow us to show asymptotic normality for a special case of the GCSA algorithm.

Note that in (5.7) we have $\boldsymbol{\Gamma} = \boldsymbol{A}\boldsymbol{H}(\theta^*)$, where $\boldsymbol{A}$ is a diagonal matrix with ith diagonal entry equal to $a^{(1)}$ if $i \leq p'$ and with ith diagonal entry equal to $a^{(2)}$ if $i > p'$. While $\boldsymbol{\Gamma} = \boldsymbol{A}\boldsymbol{H}(\theta^*)$ is not generally real symmetric and positive definite as required by Fabian's condition B1 (an exception being the case where $L(\theta)$ is linearly separable in $\theta^{(1)}$ and $\theta^{(2)}$), it is entirely possible for this matrix to have strictly positive eigenvalues and be real-diagonalizable (i.e., $\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{H}(\theta^*)\boldsymbol{S} = \boldsymbol{\Lambda}$ for a nonsingular real matrix $\boldsymbol{S}$ and a positive definite diagonal matrix $\boldsymbol{\Lambda}$); the following proposition formalizes this observation.

Proposition 1. Let $\boldsymbol{A}$ and $\boldsymbol{H}(\theta^*)$ be real square matrices. Additionally, let $\boldsymbol{A}$ be a diagonal matrix with strictly positive diagonal entries. Finally, let $\boldsymbol{H}(\theta^*)$ be symmetric and positive definite (a common assumption in minimization problems). Then, there exist a nonsingular real matrix $\boldsymbol{S}$ and a positive definite diagonal matrix $\boldsymbol{\Lambda}$ such that $\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{H}(\theta^*)\boldsymbol{S} = \boldsymbol{\Lambda}$.

Proof. Because $\boldsymbol{A}$ and $\boldsymbol{H}(\theta^*)$ are both real and positive definite, Corollary 2.5.14 in Horn and Johnson (2010) implies there exist real, symmetric, positive definite matrices $\boldsymbol{A}^{1/2}$ and $\boldsymbol{H}(\theta^*)^{1/2}$ such that $\boldsymbol{A} = \boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}$ and $\boldsymbol{H}(\theta^*) = \boldsymbol{H}(\theta^*)^{1/2}\boldsymbol{H}(\theta^*)^{1/2}$. Therefore, $\boldsymbol{A}\boldsymbol{H}(\theta^*) = \boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}\boldsymbol{H}(\theta^*)^{1/2}\boldsymbol{H}(\theta^*)^{1/2}$. By multiplying the previous equation on the right by $\boldsymbol{I} = \boldsymbol{A}^{1/2}\boldsymbol{A}^{-1/2}$, we have:

$$\boldsymbol{A}\boldsymbol{H}(\theta^*) = \boldsymbol{A}^{1/2}[\boldsymbol{H}(\theta^*)^{1/2}\boldsymbol{A}^{1/2}]^\top[\boldsymbol{H}(\theta^*)^{1/2}\boldsymbol{A}^{1/2}]\boldsymbol{A}^{-1/2}.$$

Define $\boldsymbol{M} \equiv \boldsymbol{H}(\theta^*)^{1/2}\boldsymbol{A}^{1/2}$. Then, $\boldsymbol{M}^\top\boldsymbol{M}$ is a positive definite real symmetric matrix. Thus, Corollary 2.5.14 in Horn and Johnson (2010) once again implies that we can write $\boldsymbol{M} = \boldsymbol{P}_0\Lambda\boldsymbol{P}_0^{-1}$ for a nonsingular real orthogonal matrix $\boldsymbol{P}_0$ and a diagonal matrix, Λ, with strictly positive eigenvalues. Then, $\boldsymbol{A}\boldsymbol{H}(\theta^*) = \boldsymbol{S}\Lambda\boldsymbol{S}^{-1}$ with $\boldsymbol{S} = \boldsymbol{A}^{1/2}\boldsymbol{P}_0$. Note that while $\boldsymbol{S}$ is a real matrix, it is not necessarily an orthogonal matrix (i.e., $\boldsymbol{S}^{-1}$ is not necessarily equal to $\boldsymbol{S}^\top$), this follows from the fact that $\boldsymbol{S}^{-1} = \boldsymbol{P}_0^\top\boldsymbol{A}^{-1/2}$ and $\boldsymbol{S}^\top = \boldsymbol{P}_0^\top\boldsymbol{A}^{1/2}$ so that having $\boldsymbol{S}^{-1} = \boldsymbol{S}^\top$ would imply $\boldsymbol{A}^{-1/2} = \boldsymbol{A}^{1/2}$, which is only true of $\boldsymbol{A} = \boldsymbol{I}$. □

It follows from Proposition 1 that generalizing Condition B1 in Fabian's theorem to allow Γ to be any real diagonalizable matrix with strictly positive eigenvalues (i.e., $\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{H}(\theta^*)\boldsymbol{S} = \Lambda$ with $\boldsymbol{S}$ and Λ defined as in Proposition 1) would allow for treatment of the cyclic seesaw SA algorithm. Furthermore, Section 5.4 shows that the proposed generalization to Fabian's theorem would

also allow for treatment of Algorithm 3 (deterministic pattern for coordinate selection), a special case of the GCSA algorithm (Algorithm 1).

In the following section we provide a generalization of Fabian's theorem which is slightly more general than the extension suggested by Proposition 1. The generalization only requires that $\Gamma = \boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}$ for a real non-singular matrix $\boldsymbol{S}$ and a real upper triangular matrix $\boldsymbol{U}$ with strictly positive diagonal entries (here Γ is said to be upper-triangularizable). Appendix A discusses how the generalization to Fabian's theorem expands the theorem's applicability to include other SA algorithms (aside from Algorithm 3) of practical interest.

5.3 Generalizing Fabian's Theorem

This section contains a generalization to Fabian's theorem derived by replacing condition B1 with a slightly weaker assumption. Specifically, for $\boldsymbol{W}_k$ and β defined as in (5.1) we show $k^{\beta/2}\boldsymbol{W}_k$ converges in distribution to a multivariate normal random variable under conditions B0, B2–B6, and a relaxed version of B1. Following the proof of Fabian's theorem (Fabian 1968), we begin by showing that $k^{\beta/2}\boldsymbol{W}_k$ is asymptotically normally distributed if and only if a much simpler process is also asymptotically normally distributed. After showing that the simpler process does, in fact, converge in distribution to a multivariate normal random variable, the parameters of its asymptotic distri-

bution will uniquely determine the parameters of the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$. Next we introduce some notation and the generalized version of condition B1.

Throughout this Chapter M_{ij} and $M_{k(i,j)}$ denote the (i,j)th entries of the matrices $\boldsymbol{M}$ and $\boldsymbol{M}_k$, respectively, and v_i and $v_{k(i)}$ denote the ith entries of the vectors $\boldsymbol{v}$ and $\boldsymbol{v}_k$, respectively. Furthermore, $k \geq 1$ denotes a strictly positive integer; $\xrightarrow{\text{dist}}$ means convergence in distribution; $\boldsymbol{V}_k$, $\boldsymbol{W}_k$, $\boldsymbol{T}_k$, and $\boldsymbol{T}$ are vectors in $\mathbb{R}^p$; $\boldsymbol{\Gamma}_k$, $\boldsymbol{\Phi}_k$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Phi}$, and $\boldsymbol{P}$ are matrices in $\mathbb{R}^{p\times p}$; and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{M})$ denotes a multivariate normal random variable with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{M}$. Furthermore, throughout this section we assume the recursion for $\boldsymbol{W}_k$ given in (5.1) satisfies the following conditions:

B1′ There exists an *upper triangular* matrix $\boldsymbol{U} \in \mathbb{R}^{p\times p}$ with strictly positive eigenvalues and a nonsingular $\boldsymbol{S} \in \mathbb{R}^{p\times p}$ such that $\boldsymbol{\Gamma}_k \to \boldsymbol{\Gamma} = \boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}$ w.p.1.

B6′ Define $\lambda \equiv \min_i \{U_{ii}\}$. Let α and β be constants such that $0 < \alpha \leq 1$ and $0 \leq \beta$. Define $\beta_+ \equiv \beta$ if $\alpha = 1$ and $\beta_+ \equiv 0$ otherwise. Then, $\beta_+ < 2\lambda$.

B0′ & B2′–B5′ The same as B0 and B2–B5, respectively.

Conditions B0′ and B2′–B6′ are identical to the conditions of Fabian's theorem (note that B6′ is obtained by rewriting B6 in the notation of B1′). On the other hand, B1′ is the relaxed version of Fabian's corresponding condition (condition B1 in Theorem 3) requiring symmetry of $\boldsymbol{\Gamma}$. As discussed in the comment fol-

lowing Proposition 1, any real square matrix with real eigenvalues satisfies B1′. The following Theorem is our generalization to Fabian's theorem based on conditions B1′–B6′.

Theorem 4 (A Generalization of Fabian's Theorem)**.** Assume the recursion for $\boldsymbol{W}_k$ given in (5.1) satisfies B0′–B6′. Then, the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ is a multivariate normal random variable with mean $\boldsymbol{S}\boldsymbol{\nu}$ and covariance matrix $\boldsymbol{S}\boldsymbol{Q}\boldsymbol{S}^\top$, where the entries of $\boldsymbol{\nu}$ are the unique solution to:

$$\nu_i \equiv (\tilde{U}_{ii})^{-1}\tilde{T}_i - \left[(\tilde{U}_{ii})^{-1}\sum_{j=i+1}^{p}\tilde{U}_{ij}\nu_j\right] \tag{5.8}$$

with $\tilde{\boldsymbol{T}} \equiv \boldsymbol{S}^{-1}\boldsymbol{T}$ and $\tilde{\boldsymbol{U}} \equiv \boldsymbol{U} - (\beta_+/2)\boldsymbol{I}$; and the entries of $\boldsymbol{Q}$ are the unique solution to:

$$Q_{ij} = \frac{[\boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top(\boldsymbol{S}^{-1})^\top]_{ij}}{\tilde{U}_{ii}+\tilde{U}_{jj}} - \left[\frac{\sum_{\ell=j+1}^{p}\tilde{U}_{j\ell}\,Q_{i\ell} + \sum_{\ell=i+1}^{p}\tilde{U}_{i\ell}\,Q_{\ell j}}{\tilde{U}_{ii}+\tilde{U}_{jj}}\right]. \tag{5.9}$$

Using (5.9), Q_{ij} is a function of the elements of the set $\{Q_{mn}\}_{(m,n)\in\mathcal{G}}$ where $\mathcal{G}$ is the set of (m,n) tuples such that either $m \geq i+1$ and $n \geq j+1$, $m = i$ and $n \geq j+1$, or $m \geq i+1$ and $n = j$. Therefore, the entries of $\boldsymbol{Q}$ can be computed sequentially beginning with Q_{pp}, for which (5.9) gives a solution. Similarly, the entries of $\boldsymbol{\nu}$ can be computed beginning with ν_p, for which (5.8) gives a solution.

Proof. (Although the following proof is complete, a more detailed proof of the

theorem can be found in Appendix A). In order to compute the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ we begin by constructing the following process:

$$\tilde{\boldsymbol{W}}_k = (k-1)^{\beta/2}\boldsymbol{S}^{-1}\boldsymbol{W}_k,$$

where $\boldsymbol{S}$ is the matrix from B1′. Here, using Slutsky's theorem it follows that if $\tilde{\boldsymbol{W}}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{M})$, then $k^{\beta/2}\boldsymbol{W}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{S}\boldsymbol{\mu}, \boldsymbol{S}\boldsymbol{M}\boldsymbol{S}^\top)$. Thus, proving that the process $\tilde{\boldsymbol{W}}_k$ is asymptotically normally distributed (with certain mean vector and covariance matrix) is sufficient for computing the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$. Moreover, after some algebraic manipulation it can be shown that:

$$\tilde{\boldsymbol{W}}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{\Gamma}}_k)\tilde{\boldsymbol{W}}_k + \frac{\boldsymbol{S}^{-1}\boldsymbol{T}_k}{k^\alpha} + \frac{\boldsymbol{S}^{-1}\boldsymbol{\Phi}_k\boldsymbol{V}_k}{k^{\alpha/2}}, \tag{5.10}$$

where $\tilde{\boldsymbol{W}}_1 = \boldsymbol{0}$,

$$\tilde{\boldsymbol{\Gamma}}_k \equiv \left(\frac{k}{k-1}\right)^{\beta/2}\boldsymbol{S}^{-1}\boldsymbol{\Gamma}_k\boldsymbol{S} - \left[\left(\frac{k}{k-1}\right)^{\beta/2} - 1\right]k^\alpha\boldsymbol{I},$$

and $\tilde{\boldsymbol{\Gamma}}_k \to \tilde{\boldsymbol{U}} = \boldsymbol{U} - (\beta_+/2)\boldsymbol{I}$ w.p.1 so that $\tilde{\boldsymbol{W}}_k$ is a special case of (5.1). Using the same arguments as those in Fabian (1968, proof of Theorem 2.2) it can be shown that replacing $\boldsymbol{T}_k$ with $\boldsymbol{T}$ and $\boldsymbol{\Phi}_k$ with $\boldsymbol{\Phi}$ in (5.10) does not change the asymptotic distribution of $\tilde{\boldsymbol{W}}_k$ (this result is not immediate due to the recursive nature of $\tilde{\boldsymbol{W}}_k$). Therefore, in order to show that $\tilde{\boldsymbol{W}}_k$ is asymptotically normally

distributed we may assume, without loss of generality (w.l.o.g.), that $\boldsymbol{T}_k = \boldsymbol{T}$ and $\boldsymbol{\Phi}_k = \boldsymbol{\Phi}$ so that:

$$\tilde{\boldsymbol{W}}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{\Gamma}}_k)\tilde{\boldsymbol{W}}_k + k^{-\alpha}\tilde{\boldsymbol{T}} + k^{-\alpha/2}\tilde{\boldsymbol{V}}_k, \tag{5.11}$$

where $\tilde{\boldsymbol{T}} = \boldsymbol{S}^{-1}\boldsymbol{T}$ and $\tilde{\boldsymbol{V}}_k \equiv \boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{V}_k$. Next, we relate the asymptotic distribution of $\tilde{\boldsymbol{W}}_k$, as described by (5.11), to that of an even simpler process.

Consider the process:

$$\tilde{\boldsymbol{W}}'_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{\Gamma}}_k)\tilde{\boldsymbol{W}}'_k + k^{-\alpha/2}\tilde{\boldsymbol{V}}_k, \tag{5.12}$$

obtained by removing the term $k^{-\alpha}\tilde{\boldsymbol{T}}$ from (5.11). Lemma 4.2 (Fabian 1967) implies $\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k \to \boldsymbol{\nu}$ w.p.1, where the entries of $\boldsymbol{\nu}$ are given in (5.8). The significance of this observation is that if $\tilde{\boldsymbol{W}}'_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{M})$, then $\tilde{\boldsymbol{W}}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\nu}, \boldsymbol{M})$. At this point, the same arguments as those in Fabian (1968, proof of Theorem 2.2) can be used to show the following two results:

1. Replacing $\tilde{\boldsymbol{\Gamma}}_k$ with $\tilde{\boldsymbol{U}}$ in (5.12) does not change the asymptotic distribution of $\tilde{\boldsymbol{W}}'_k$ which depends on the limit (w.p.1) of $\tilde{\boldsymbol{\Gamma}}_k$ but not on $\tilde{\boldsymbol{\Gamma}}_k$ itself. Therefore, w.l.o.g. we may assume that $\tilde{\boldsymbol{\Gamma}}_k = \tilde{\boldsymbol{U}}$ so that:

$$\tilde{\boldsymbol{W}}'_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{U}})\tilde{\boldsymbol{W}}'_k + k^{-\alpha/2}\tilde{\boldsymbol{V}}_k. \tag{5.13}$$

2. The characteristic function of the asymptotic distribution of $\tilde{\boldsymbol{W}}_k'$ evaluated at $\boldsymbol{t} \in \mathbb{R}^p$ depends on $\boldsymbol{t}$, α, $\tilde{\boldsymbol{U}}$, and on $\tilde{\boldsymbol{\Sigma}} \equiv \lim_{k\to\infty} \mathrm{cov}[\tilde{\boldsymbol{V}}_k|\mathcal{F}_k]$ but is independent of other aspects of the distribution of $\tilde{\boldsymbol{V}}_k$ (provided conditions B0′–B6′ hold). Therefore, we may assume w.l.o.g. that the vectors $\tilde{\boldsymbol{V}}_k$ are i.i.d. $\mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Sigma}})$.

Next we derive the asymptotic distribution of the process $\tilde{\boldsymbol{W}}_k'$ in (5.13).

First, note that $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top(\boldsymbol{S}^{-1})^\top$ by condition B4′. Next, by point 2 in the previous paragraph we may assume w.l.o.g. that the vectors $\tilde{\boldsymbol{V}}_k$ are i.i.d. $\mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Sigma}})$. Consequently, the distribution of $\tilde{\boldsymbol{W}}_k'$ from (5.13) must be a multivariate normal random variable with mean zero (since $\tilde{\boldsymbol{W}}_1' = \boldsymbol{0}$) and covariance $\boldsymbol{Q}_k \equiv E[\tilde{\boldsymbol{W}}_k'(\tilde{\boldsymbol{W}}_k')^\top]$ which satisfies:

$$\begin{aligned} Q_{k+1(ij)} = &\left(1 - k^{-\alpha}\left[\tilde{U}_{ii} + \tilde{U}_{jj} - k^{-\alpha}\tilde{U}_{ii}\,\tilde{U}_{jj}\right]\right) Q_{k(ij)} \\ &- k^{-\alpha}\left[\sum_{\ell=j+1}^{p} \tilde{U}_{j\ell}\, Q_{k(i\ell)} + \sum_{\ell=i+1}^{p} \tilde{U}_{i\ell}\, Q_{k(\ell j)}\right] + k^{-\alpha}\tilde{\Sigma}_{ij} \\ &+ k^{-2\alpha}\sum_{\ell=i}^{p} \tilde{U}_{i\ell} \sum_{s=j+1}^{p} \tilde{U}_{js}\, Q_{\ell s} + k^{-2\alpha}\sum_{\ell=i+1}^{p} \tilde{U}_{i\ell}\,\tilde{U}_{jj}\, Q_{\ell j}. \end{aligned}$$

Here, Lemma 4.2 in Fabian (1967) implies $\boldsymbol{Q} \equiv \lim_{k\to\infty} \boldsymbol{Q}_k$ is a matrix whose entries are the unique solution to (5.9). Therefore, $\tilde{\boldsymbol{W}}_k' \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$ which implies $\boldsymbol{W}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{S}\boldsymbol{\nu}, \boldsymbol{S}\boldsymbol{Q}\boldsymbol{S}^\top)$. □

Note that if $\boldsymbol{\Gamma}$ is real symmetric then the terms in square brackets in (5.8)

and (5.9) disappear since $\tilde{U}$ may be taken to be a diagonal matrix. This implies that when Γ is a real and symmetric matrix (as in Fabian's theorem) then:

$$\boldsymbol{S}\boldsymbol{\nu} = \boldsymbol{S}\tilde{\boldsymbol{U}}^{-1}\tilde{\boldsymbol{T}} = \boldsymbol{S}[\boldsymbol{U} - (\beta_+/2)\boldsymbol{I}]^{-1}\boldsymbol{S}^{-1}\boldsymbol{T} = (\boldsymbol{\Gamma} - (\beta_+/2)\boldsymbol{I})^{-1}\boldsymbol{T}. \tag{5.14}$$

The mean vector in (5.14) is the same as the mean of the asymptotic distribution in Fabian's theorem (Theorem 3). Similarly, when $\boldsymbol{U}$ is a diagonal matrix, we can assume $\boldsymbol{S} = \boldsymbol{P}$ for some orthogonal matrix $\boldsymbol{P}$ and (5.9) reduces to $Q_{ij} = (\boldsymbol{P}^\top \boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top \boldsymbol{P})_{ij}(U_{ii} + U_{jj} - \beta_+)^{-1}$ as in Fabian's theorem.

5.4 Asymptotic Normality of Algorithm 3

This section gives a set of conditions for the asymptotic normality of the normalized iterates from Algorithm 3 (a special case of GCSA). First, following a derivation similar to that of (5.5), Algorithm 3 may be written as:

$$\begin{aligned}
\hat{\theta}_{k+1} = \hat{\theta}_k &- \sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{J}^{(j(z))}(\overline{\theta}_k)(\hat{\theta}_k - \theta^*) \\
&+ \sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{J}^{(j(z))}\left(\overline{\theta}_k^{(I_{z,\ell})}\right)[\boldsymbol{\Delta}_{\boldsymbol{g}}^{(I_{z,\ell})} + \boldsymbol{\Delta}_{\boldsymbol{\beta}}^{(I_{z,\ell})} + \boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(I_{z,\ell})}] \\
&- \sum_{z=1}^{s}\sum_{\ell=0}^{n_(z)-1} a_k^{(j(z))} \boldsymbol{\beta}_k^{(j(z))}\left(\hat{\theta}_k^{(I_{z,\ell})}\right) \\
&- \sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{\xi}_k^{(j(z))}\left(\hat{\theta}_k^{(I_{z,\ell})}\right),
\end{aligned} \tag{5.15}$$

where $\hat{\theta}_k$ denotes an iterate of Algorithm 3 and

$$\boldsymbol{\Delta}_{\boldsymbol{g}}^{(I_z,\ell)} \equiv \sum_{m=1}^{z-1}\sum_{i=0}^{n(m)-1}\left[a_k^{(j(m))}\boldsymbol{g}^{(j(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right)\right] + \sum_{i=0}^{\ell-1}\left[a_k^{(j(z))}\boldsymbol{g}^{(j(z))}\left(\hat{\theta}_k^{(I_{z,i})}\right)\right], \quad (5.16a)$$

$$\boldsymbol{\Delta}_{\beta}^{(I_z,\ell)} \equiv \sum_{m=1}^{z-1}\sum_{i=0}^{n_k(m)-1}\left[a_k^{(j(m))}\boldsymbol{\beta}_k^{(j(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right)\right] + \sum_{i=0}^{\ell-1}\left[a_k^{(j(z))}\boldsymbol{\beta}_k^{(j(z))}\left(\hat{\theta}_k^{(I_{z,i})}\right)\right], \quad (5.16b)$$

$$\boldsymbol{\Delta}_{\xi}^{(I_z,\ell)} \equiv \sum_{m=1}^{z-1}\sum_{i=0}^{n_k(m)-1}\left[a_k^{(j(m))}\boldsymbol{\xi}_k^{(j(m))}\left(\hat{\theta}_k^{(I_{m,i})}\right)\right] + \sum_{i=0}^{\ell-1}\left[a_k^{(j(z))}\boldsymbol{\xi}_k^{(j(z))}\left(\hat{\theta}_k^{(I_{z,i})}\right)\right]. \quad (5.16c)$$

Then, in a manner analogous to (5.6a–c) we let $\alpha > 0$, $\beta \geq 0$, $\boldsymbol{W}_k = \hat{\theta}_k = \theta^*$, and let $\mathcal{F}_k$ be the sigma field generated by $\{\hat{\theta}_\ell\}_{\ell=0}^k$ as well as by any random variables generated by the algorithm in the production of $\hat{\theta}_k$. Additionally, let

$$\begin{aligned}\boldsymbol{T}_k = & -k^{\alpha+\beta/2}\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))}\boldsymbol{\beta}_k^{(j(z))}\left(\hat{\theta}_k^{(I_z,\ell)}\right) \\ & + k^{\alpha+\beta/2}\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))}\boldsymbol{J}^{(j(z))}\left(\overline{\theta}_k^{(I_z,\ell)}\right)[\boldsymbol{\Delta}_{\boldsymbol{g}}^{(I_z,\ell)} + \boldsymbol{\Delta}_{\beta}^{(I_z,\ell)}] \\ & + k^{\alpha+\beta/2}\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))}E\left[\boldsymbol{J}^{(j(z))}\left(\overline{\theta}_k^{(I_z,\ell)}\right)\boldsymbol{\Delta}_{\xi}^{(I_z,\ell)}\middle|\mathcal{F}_k\right], \end{aligned} \quad (5.17)$$

let $\boldsymbol{\Phi}_k = \boldsymbol{I}$, let $\boldsymbol{V}_k$ be given by

$$\begin{aligned}\boldsymbol{V}_k = & -k^{(\alpha+\beta)/2}\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))}\boldsymbol{\xi}_k^{(j(z))}\left(\hat{\theta}_k^{(I_z,\ell)}\right) \\ & + k^{(\alpha+\beta)/2}\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))}\boldsymbol{J}^{(j(z))}\left(\overline{\theta}_k^{(I_z,\ell)}\right)\boldsymbol{\Delta}_{\xi}^{(I_z,\ell)} \\ & - k^{(\alpha+\beta)/2}\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} a_k^{(j(z))}E\left[\boldsymbol{J}^{(j(z))}\left(\overline{\theta}_k^{(I_z,\ell)}\right)\boldsymbol{\Delta}_{\xi}^{(I_z,\ell)}\middle|\mathcal{F}_k\right], \end{aligned} \quad (5.18)$$

and let

$$\boldsymbol{\Gamma}_k = k^{\alpha} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{J}^{(j(z))}(\overline{\theta}_k). \tag{5.19}$$

With this notation, the recursion in (5.15) defining an iteration of Algorithm 3 can be written in the form of (5.1), the recursion of Theorem 4.

Because (5.15) is a special case of (5.1), conditions B0′–B6′ would be sufficient for the asymptotic normality of $k^{\beta/2}(\hat{\theta}_k - \theta^*)$ when $\hat{\theta}_k$ denotes an iterate of Algorithm 3. In practice, however, the validity of conditions B0′–B6′ (or of the slightly stronger conditions B0–B6) can be difficult to verify for *any* algorithm which may be written in the form of (5.1). In the case of Algorithm 3 this is complicated by the fact that the terms $\boldsymbol{\Gamma}_k$, $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{V}_k$ have more complex forms than the corresponding variables for classical SA algorithms (compare the definitions below (5.3) to (5.17)–(5.19)). In an effort to obtain a set of conditions that are easier to understand than B0′–B6′, this section derives a set of conditions which imply B0′–B6′ hold for Algorithm 3. To do this, we first present a useful lemma related to condition B5′.

Lemma 13. Let $\boldsymbol{V}_k$ be a vector-valued sequence with $\boldsymbol{V}_k = \sum_{i=1}^{N} \boldsymbol{\Psi}_k(i)$, where N is an integer such that $2 \leq N < \infty$ and $\boldsymbol{\Psi}_k(i)$ denotes a vector-valued random variable. Additionally, for some $\alpha > 0$ and for every $r > 0$ let the values $\sigma_{k,r}^2(i) \equiv E\chi\{\|\boldsymbol{\Psi}_k(i)\|^2 \geq rk^{\alpha}\}\|\boldsymbol{\Psi}_k(i)\|^2$ satisfy $\lim_{k\to\infty} \sigma_{k,r}^2(i) = 0$ for all i. Then,

the sequence $\sigma_{k,r}^2 \equiv E\chi\{\|\boldsymbol{V}_k\|^2 \geq rk^\alpha\}\|\boldsymbol{V}_k\|^2$ also satisfies $\lim_{k\to\infty}\sigma_{k,r}^2 = 0$.

Proof. For simplicity, we assume $N = 2$ so that $\boldsymbol{V}_k = \boldsymbol{\Psi}_k(1) + \boldsymbol{\Psi}_k(2)$ (the proof for $N > 2$ is essentially identical). First, using the triangle inequality we know $\|\boldsymbol{V}_k\| \leq \|\boldsymbol{\Psi}_k(1)\| + \|\boldsymbol{\Psi}_k(2)\|$. Therefore, in order to have $\|\boldsymbol{V}_k\| \geq \sqrt{rk^\alpha}$ at least one of $\|\boldsymbol{\Psi}_k(1)\|$ or $\|\boldsymbol{\Psi}_k(2)\|$ must be greater or equal to $\sqrt{rk^\alpha}/2$, which implies:

$$\chi\{\|\boldsymbol{V}_k\| \geq \sqrt{rk^\alpha}\}\|\boldsymbol{V}_k\| \leq \sum_{i=1}^{2} 2\chi\{\|\boldsymbol{\Psi}_k(i)\| \geq \sqrt{rk^\alpha}/2\}\|\boldsymbol{\Psi}_k(i)\|$$

(recall that $\chi\{\mathcal{E}\}$ is the indicator function of the event $\mathcal{E}$). Equivalently,

$$\chi\{\|\boldsymbol{V}_k\|^2 \geq rk^\alpha\}\|\boldsymbol{V}_k\| \leq \sum_{i=1}^{2} 2\chi\{\|\boldsymbol{\Psi}_k(i)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(i)\|. \tag{5.20}$$

Taking the square on both sides of (5.20) gives:

$$\begin{aligned}\chi\{\|\boldsymbol{V}_k\|^2 \geq rk^\alpha\}\|\boldsymbol{V}_k\|^2 &\leq \sum_{i=1}^{2} 4\chi\{\|\boldsymbol{\Psi}_k(i)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(i)\|^2 \\ &+ 8\chi\{\|\boldsymbol{\Psi}_k(1)\|^2 \geq rk^\alpha/4\}\chi\{\|\boldsymbol{\Psi}_k(2)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(1)\|\|\boldsymbol{\Psi}_k(2)\|. \end{aligned}\tag{5.21}$$

Taking the expectation on both sides of (5.21) and using the definition of $\sigma_{k,r}^2$:

$$\begin{aligned}\sigma_{k,r}^2 &\leq \sum_{i=1}^{2} 4E\Big[\chi\{\|\boldsymbol{\Psi}_k(i)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(i)\|^2\Big] \\ &+ 8E\Big[\chi\{\|\boldsymbol{\Psi}_k(1)\|^2 \geq rk^\alpha/4\}\chi\{\|\boldsymbol{\Psi}_k(2)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(1)\|\|\boldsymbol{\Psi}_k(2)\|\Big]. \end{aligned}\tag{5.22}$$

By assumption, however, $\lim_{k\to\infty} E\left[\chi\{\|\boldsymbol{\Psi}_k(i)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(i)\|^2\right] = 0$. Therefore, (5.22) becomes:

$$\sigma_{k,r}^2 \leq o(1) + 8E\Big[\chi\{\|\boldsymbol{\Psi}_k(1)\|^2 \geq rk^\alpha/4\}\chi\{\|\boldsymbol{\Psi}_k(2)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(1)\|\|\boldsymbol{\Psi}_k(2)\|\Big], \tag{5.23}$$

where $o(\cdot)$ denotes the standard little-o notation. Next, using the Cauchy–Schwarz inequality:

$$E\Big[\chi\{\|\boldsymbol{\Psi}_k(1)\|^2 \geq rk^\alpha/4\}\chi\{\|\boldsymbol{\Psi}_k(2)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(1)\|\|\boldsymbol{\Psi}_k(2)\|\Big] \leq$$
$$\Big(E\Big[\chi\{\|\boldsymbol{\Psi}_k(1)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(1)\|^2\Big]\Big)^{1/2}\Big(E\Big[\chi\{\|\boldsymbol{\Psi}_k(2)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(2)\|^2\Big]\Big)^{1/2}. \tag{5.24}$$

Once again, since $\lim_{k\to\infty} E\left[\chi\{\|\boldsymbol{\Psi}_k(i)\|^2 \geq rk^\alpha/4\}\|\boldsymbol{\Psi}_k(i)\|^2\right] = 0$, combining (5.23) and (5.24) implies $\sigma_{k,r}^2 = o(1)$, as desired. □

The following theorem gives conditions for the asymptotic normality of the normalized iterates from Algorithm 3.

Theorem 5. Let $\hat{\theta}_k$ be generated according to Algorithm 3. Additionally, let $\mathcal{F}_k$ denote the sigma field generated by $\{\hat{\theta}_\ell\}_{\ell=0}^k$ as well as by any random variables generated by the algorithm in the production of $\hat{\theta}_k$. Let $\boldsymbol{W}_k = \hat{\theta}_k - \theta^*$, let β, α, $\boldsymbol{\Gamma}_k$, $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{V}_k$ be defined as in (5.17)–(5.19) and assume these variables have real-valued entries. Additionally, assume the following conditions hold:

C0 $L(\theta)$ is twice continuously differentiable, $\boldsymbol{H}(\theta)$ (the Hessian of $L(\theta)$) has bounded entries (i.e., $\boldsymbol{g}(\theta)$ is Lipschitz continuous), and $\hat{\theta}_k^{(I_{z,i})} \to \theta^*$ w.p.1 uniformly over z and i.

C1 For each j, $a_k^{(j)} > 0$ and $k^\alpha a_k^{(j)} \to r_j < \infty$.

C2 There exists an upper triangular matrix $\boldsymbol{U} \in \mathbb{R}^{p\times p}$ with strictly positive eigenvalues and a nonsingular matrix $\boldsymbol{S} \in \mathbb{R}^{p\times p}$ such that

$$\Gamma \equiv \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} r_{j(z)} \boldsymbol{J}^{(j(z))}(\theta^*) = \boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}, \tag{5.25}$$

where $\boldsymbol{J}^{(j)}(\theta)$ denotes the Jacobian of $\boldsymbol{g}^{(j)}(\theta)$.

C3 β and α satisfy $0 < \beta \leq \alpha$. Additionally, for all j there exist a finite vector $\boldsymbol{b}^{(j)} \in \mathbb{R}^p$ such that $k^{\beta/2}\beta_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big) \to \boldsymbol{b}^{(j(z))}$ w.p.1 uniformly over z and i.

C4 For all z and i, $E\left[\xi_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)\middle|\hat{\theta}_k^{(I_{z,i})}\right] = \boldsymbol{0}$.

C5 For all z there exists a constant $C > 0$ and a matrix $\Sigma^{(j)}$ such that one of the following holds w.p.1 (convergence is meant uniformly over i):

(i) $\beta = \alpha$, $C > \|E[\xi_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)[\xi_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)]^\top|\mathcal{F}_k] - \Sigma^{(j(z))}\| \to 0$.

(ii) $\beta < \alpha$, $C > \|k^{\beta-\alpha}E[\xi_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)[\xi_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)]^\top|\mathcal{F}_k] - \Sigma^{(j(z))}\| \to 0$.

Additionally, if $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two different elements of $\{\xi_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)\}_{z,i}$, then one of the following holds w.p.1:

(iii) C5-(i) holds and $C > \|E[\boldsymbol{v}_1 \boldsymbol{v}_2^\top | \mathcal{F}_k]\| \to 0$.

(iv) C5-(ii) holds and $C > \|k^{\beta-\alpha} E[\boldsymbol{v}_1 \boldsymbol{v}_2^\top | \mathcal{F}_k]\| \to 0$.

C6 $\boldsymbol{\Sigma} \equiv \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} r_{j(z)}^2 \boldsymbol{\Sigma}^{(j(z))}$ is a positive definite matrix.

C7 For each $z = 1, \ldots, s$ and $i = 1, \ldots, n(z)$ one of the following holds:

(i) $\beta = \alpha$ and B5-(i) holds with $\boldsymbol{V}_k$ replaced by $\boldsymbol{\xi}_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)$.

(ii) $\beta < \alpha$ and B5-(i) holds with $\boldsymbol{V}_k$ replaced by $k^{(\beta-\alpha)/2}\boldsymbol{\xi}_k^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big)$.

C8 The same as B6′.

Then, the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ is a multivariate normal random variable with mean $\boldsymbol{S}\boldsymbol{\nu}$ and covariance matrix $\boldsymbol{S}\boldsymbol{Q}\boldsymbol{S}^\top$, where the entries of $\boldsymbol{\nu}$ are the unique solution to (5.8) with $\tilde{\boldsymbol{T}} = -\boldsymbol{S}^{-1}\left[\sum_{z=1}^{s}\sum_{\ell=0}^{n_(z)-1} r_{j(z)} \boldsymbol{b}^{(j(z))}\right]$ and $\tilde{\boldsymbol{U}} = \boldsymbol{U} - (\beta_+/2)\boldsymbol{I}$, and the entries of $\boldsymbol{Q}$ are the unique solution to (5.9) with $\tilde{\boldsymbol{U}}$ defined as above and $\boldsymbol{\Phi} = \boldsymbol{I}$.

Proof. We show that conditions B0′–B6′ hold. The result then follows from Theorem 4. First, note that condition B0′ holds by the definition of $\boldsymbol{\Phi}_k$, $\boldsymbol{\Gamma}_k$, $\boldsymbol{V}_k$, and $\mathcal{F}_k$. Next, we show that B1′ holds. First, by C0 and C1:

$$\lim_{k\to\infty} \boldsymbol{\Gamma}_k = \lim_{k\to\infty} k^{\alpha} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{J}^{(j(z))}(\overline{\theta}_k) = \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} r_{j(z)} \boldsymbol{J}^{j(z)}(\theta^*) \text{ w.p.1.}$$

Then, C2 guarantees B1′ holds with $\boldsymbol{\Gamma}$ equal to the matrix in (5.25). The fact

that B2′ holds follows immediately from the fact that $\Phi_k = \boldsymbol{I}$ for all k, so that Φ in B1′ is equal to the identity matrix. Next we show that B3′ holds.

First, using C1 and C3 it follows that

$$-\lim_{k\to\infty} k^{\alpha+\beta/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{\beta}_k^{(j(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) = -\sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} r_{j(z)} \boldsymbol{b}^{(j(z))} \text{ w.p.1.} \quad (5.26)$$

Next, C0, C1, and C3 imply that

$$\begin{aligned} &\lim_{k\to\infty} k^{\alpha+\beta/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{J}^{(j(z))} \left(\overline{\theta}_k^{(I_{z,\ell})} \right) [\boldsymbol{\Delta}_{\boldsymbol{g}}^{(I_{z,\ell})} + \boldsymbol{\Delta}_{\boldsymbol{\beta}}^{(I_{z,\ell})}] \\ &= \lim_{k\to\infty} k^{\alpha+\beta/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} O(k^{\alpha}) \boldsymbol{J}^{(j(z))} \left(\overline{\theta}_k^{(I_{z,\ell})} \right) o(k^{\alpha}) \\ &= \mathbf{0} \text{ w.p.1.} \end{aligned} \quad (5.27)$$

Lastly, by C0 (the boundedness of $\boldsymbol{H}(\theta)$), C1, C3 (using $0 < \beta \leq \alpha$), and C5:

$$\begin{aligned} &\lim_{k\to\infty} k^{\alpha+\beta/2} \left\| \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} E\left[\boldsymbol{J}^{(j(z))} \left(\overline{\theta}_k^{(I_{z,\ell})} \right) \boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(I_{z,\ell})} \middle| \mathcal{F}_k \right] \right\| \\ &\leq \lim_{k\to\infty} k^{\alpha+\beta/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \left(E\left[\left\| \boldsymbol{J}^{(j(z))} \left(\overline{\theta}_k^{(I_{z,\ell})} \right) \boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(I_{z,\ell})} \right\|^2 \middle| \mathcal{F}_k \right] \right)^{1/2} \\ &\leq \lim_{k\to\infty} k^{\alpha+\beta/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} O(k^{\alpha}) \left(E\left[\left\| \boldsymbol{J}^{(j(z))} \left(\overline{\theta}_k^{(I_{z,\ell})} \right) \right\|^2 \left\| \boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(I_{z,\ell})} \right\|^2 \middle| \mathcal{F}_k \right] \right)^{1/2} \\ &\leq \lim_{k\to\infty} k^{\alpha+\beta/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} O(k^{\alpha}) \left[E\left\| \boldsymbol{\Delta}_{\boldsymbol{\xi}}^{(I_{z,\ell})} \right\|^2 \middle| \mathcal{F}_k \right]^{1/2} \\ &= 0 \text{ w.p.1}, \end{aligned} \quad (5.28)$$

where the first inequality in (5.28) follows from the triangle inequality and the fact that $\|E[\boldsymbol{v}]\|^2 \leq E\|\boldsymbol{v}\|^2$ for $\boldsymbol{v} \in \mathbb{R}^p$ so that $\|E[\boldsymbol{v}]\| \leq \sqrt{E\|\boldsymbol{v}\|^2}$, the third inequality follows from the submultplicativity of the Frobenius norm ($\|\cdot\|$ applied to a matrix is the Frobenius norm), and the last inequality follows from the boundedness of the the Hessian (which implies the boundedness of $\boldsymbol{J}^{(j)}(\theta)$). Thus, the definition of $\boldsymbol{T}_k$ in (5.17) along with (5.26)–(5.28) imply

$$\boldsymbol{T}_k \to -\sum_{z=1}^{s} \sum_{\ell=0}^{n_(z)-1} r_{j(z)} \boldsymbol{b}^{(j(z))} \text{ w.p.1.} \tag{5.29}$$

This implies B3′ holds with $\boldsymbol{T}$ defined as in the right-hand side of (5.29). Next we show B4′ holds.

That $E[\boldsymbol{V}_k|\mathcal{F}_k] = \mathbf{0}$ follows immediately from C4. Next we show that there exists a constant $C' < \infty$ such that $C' > \|E[\boldsymbol{V}_k\boldsymbol{V}_k^\top|\mathcal{F}_k] - \boldsymbol{\Sigma}\| \to 0$, where $\boldsymbol{\Sigma}$ is as in C6. First, by C1, C4, and C5:

$$C_1 > \left\| \operatorname{Var}\left[k^{(\alpha+\beta)/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{\xi}_k^{(j(z))}\left(\hat{\boldsymbol{\theta}}_k^{\left(I_{z,\ell}\right)}\right) \middle| \mathcal{F}_k \right] - \boldsymbol{\Sigma} \right\| \to 0, \tag{5.30}$$

for some constant $C_1 < \infty$. (Note that the vector inside the variance in the previous equation has mean zero by the first part of condition C4. Therefore, its variance is equal to its second moment matrix.) Next, by the definition of $\boldsymbol{\Delta}_{\xi}^{(\cdot)}$ in (5.16c) and using conditions C0, C1, and C5 it follows that there must

exist a constant $C_2 < \infty$ such that:

$$C_2 > \left\| \text{Var} \left[k^{(\alpha+\beta)/2} \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} a_k^{(j(z))} \boldsymbol{J}^{(j(z))} \left(\overline{\theta}_k^{(I_{z,\ell})} \right) \boldsymbol{\Delta}_{\xi}^{(I_{z,\ell})} \middle| \mathcal{F}_k \right] \right\| \to 0. \tag{5.31}$$

From (5.30) we see that the second moment matrix of the first line of (5.18) is bounded and converges to $\boldsymbol{\Sigma}$. Additionally, (5.31) says that the second moment matrix of the last two lines of (5.18) is bounded and converges to the zero matrix. Furthermore, any cross-term expectation involving a summand from the first line of (5.18) and a summand from the second or third line of (5.18) is also bounded and converges to the zero matrix due to the fact that:

$$k^{\beta-\alpha} O(k^{-\alpha}) \left(E \left\| \xi_k^{(j(z))} \left(\hat{\theta}_k^{(I_{z,\ell})} \right) \middle| \mathcal{F}_k \right\|^2 \right)^{1/2} \left(E \left\| \xi_k^{(j(m))} \left(\hat{\theta}_k^{(I_{m,i})} \right) \middle| \mathcal{F}_k \right\|^2 \right)^{1/2} = o(1),$$

a consequence of C5. In conclusion, $E[\boldsymbol{V}_k \boldsymbol{V}_k^\top | \mathcal{F}_k]$ is equal to the sum of finite number of bounded matrices, most of which converge to the zero matrix w.p.1 except for one matrix (the matrix in the variance of equation 5.30) which converges to $\boldsymbol{\Sigma}$ w.p.1. Therefore, there exists a constant $C' < \infty$ such that $C' > \| E[\boldsymbol{V}_k \boldsymbol{V}_k^\top | \mathcal{F}_k] - \boldsymbol{\Sigma} \| \to 0$ and B4′ holds.

It remains to show that B5′ and B6′ hold. First, conditions C0 and C7 imply $\boldsymbol{V}_k$ satisfies the conditions of Lemma 13. Therefore, by Lemma 13 the vector $\boldsymbol{V}_k$ satisfies B5-(i) (a special case of B5′). Lastly, since C8 and B6′ are identical, condition B6′ is automatically satisfied. Since we have shown that conditions

C0–C8 imply B0′–B6′ are satisfied, the conclusion of Theorem 4 holds and the desired result follows from said theorem. □

5.5 On the Conditions for Normality

Theorem 5 stated a set of conditions for the asymptotic normality of the scaled iterates from Algorithm 3 (conditions C0–C8). This section discusses the validity of these conditions.

Condition C0. One of the assumptions of this condition is that the Hessian of $L(\theta)$, $\boldsymbol{H}(\theta)$, must be bounded uniformly over θ. This is certainly a stronger assumption than condition A0 which only requires $L(\theta)$ to be twice-continuously differentiable. Condition C0 also assumes that $\hat{\theta}_k^{(I_{z,i})} \to \theta^*$ w.p.1 uniformly over z and i, this assumption is satisfied under the conditions of Theorem 2 (see Corollary 1).

Condition C1. Note that this condition is a special case of the assumption from A0″ requiring $a_k^{(j)}/a_k \to r_j$ (condition C1 replaces a_k with $k^{-\alpha}$). As discussed in Section 4.5, C1 would be satisfied by sequences of the form $a_k^{(j)} = a^{(j)}/(1 + k + A^{(j)})^{\alpha}$ where $a^{(j)}$ and $A^{(j)}$ are both strictly-positive constants.

Condition C2. This condition requires $\Gamma = \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} r_{j(z)} \boldsymbol{J}^{(j(z))}(\theta^*)$ to be equal to $\boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}$ where $\boldsymbol{S}$ and $\boldsymbol{U}$ are both real matrices and $\boldsymbol{U}$ is positive-definite and upper-triangular. In order to discuss whether this is a reasonable

assumption, note that $\boldsymbol{J}^{(j)}(\theta^*) \in \mathbb{R}^{p\times p}$ is a matrix formed by taking $\boldsymbol{H}(\theta^*)$ and replacing the rows in the set $\mathcal{S}_j$, as defined in (3.19), with zeros. If $\boldsymbol{H}(\theta^*)$ is positive definite, one situation in which C2 holds is when $r_j > 0$ for all j and the $\mathcal{S}_j$ are disjoint sets (it is important to note that having this sets be disjoint is not a necessary condition). Then,

$$\sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1} r_{j(z)}\boldsymbol{J}^{(j(z))}(\theta^*) = \sum_{j=1}^{d} r_j d_j \boldsymbol{J}^{(j)}(\theta^*) \tag{5.32}$$

where $d_j = \sum_{z=1}^{s}\sum_{\ell=0}^{n(z)-1}\chi\{j(z)=j\}$. The variable d_j represents the (deterministic) number of times the jth subvector is updated during the kth iteration. Provided $r_j d_j > 0$, (5.32) can be rewritten as follows:

$$\sum_{j=1}^{d} r_j d_j \boldsymbol{J}^{(j)}(\theta^*) = \boldsymbol{\Lambda}\boldsymbol{H}(\theta^*), \tag{5.33}$$

where the matrix $\boldsymbol{\Lambda}$ is a positive definite diagonal matrix with ith diagonal entry equal to $r_j d_j$ where j satisfies $i \in \mathcal{S}_j$ (recall that in this example i lies in $\mathcal{S}_j$ for exactly one j since the sets $\mathcal{S}_j$ are disjoint). Then, as proven in Proposition 1, condition C2 holds. Note that having $r_j d_j > 0$ implies that $k^{\alpha}a_k^{(j)} \to r_j > 0$ and that each subvector must be updated at least once per iteration.

Condition C3. In the case where the noisy gradient estimates in Algorithm 3 are obtained in using the SG estimates (see, for example, Section 3.2) the bias term is equal to zero. Therefore, in this case we can assume $\boldsymbol{\beta}^{(j(z))}\big(\hat{\theta}_k^{(I_{z,i})}\big) = \mathbf{0}$ so

that C3 holds with $\boldsymbol{b}^{(j(z))} = 0$. When the noisy gradient estimates are obtained in an SPSA manner (see, for example, Section 3.3) the bias terms $\boldsymbol{\beta}^{(j(z))}\big(\hat{\boldsymbol{\theta}}_k^{(I_{z,i})}\big)$ cannot be assumed to be equal to the zero vector. However, analogous conditions to those of Proposition 2 in Spall (1992) (after an natural adaptation for the setting of Algorithm 3) can be used to show that the bias terms satisfy C3 when the noisy gradient updates are obtained in the manner of (3.8)–(3.9) or in the manner of (3.10)–(3.11).

Condition C4. The mean-zero assumption of the noise terms is a common assumption throughout the SA literature. The assumption essentially requires the conditional mean of the noise term (conditional on $\hat{\boldsymbol{\theta}}_k^{(I_{z,i})}$) to be absorbed into the bias term $\boldsymbol{\beta}_k^{(j(z))}\big(\hat{\boldsymbol{\theta}}_k^{(I_{z,i})}\big)$. Condition C3 would then require the conditional mean of the noise terms to converge to the zero vector.

Condition C5. Condition C5 allows for two possible scenarios. The first case, C5-(i), requires the noise vectors to have a bounded conditional covariance matrix which converges w.p.1. A simple scenario in which this holds is when the noise vectors are based on independent multivariate Gaussians with mean vector equal to zero and covariance matrix equal to $\boldsymbol{I}$. Condition C5-(ii), on the other hand, allows the noise terms to have a growing (in magnitude) conditional covariance matrix. The amount of growth, however, must be proportional to $k^{\alpha-\beta}$. This condition is well suited for algorithms such as SPSA where the conditional covariance of the noise grows as $k \to \infty$ (e.g., Spall 1992).

Conditions C5-(iii) and C5-(iv) are similar to C5-(i) and C5-(ii), respectively. Loosely speaking, C5-(iii) requires any two different noise vectors to be asymptotically uncorrelated while C5-(iv) allows the magnitude of the correlation to increase at a rate proportional to $k^{\alpha-\beta}$. One scenario where conditions C5-(iii) and C5-(iv) are satisfied is when the noise terms are independent of each other. Section 7.2.2 gives an example of where the noise terms are not independent but where C5-(iv) holds.

Condition C6. This condition requires $\mathbf{\Sigma} = \sum_{z=1}^{s} \sum_{\ell=0}^{n(z)-1} r_{j(z)} \mathbf{\Sigma}^{(j(z))}$ to be positive definite. Let us give an example of when this condition may hold. If the sets $\mathcal{S}_j$ are disjoint (i.e., that the subvectors to update do not overlap) then with d_j defined as in (5.33):

$$\mathbf{\Sigma} = \sum_{j=1}^{d} r_j d_j \mathbf{\Sigma}^{(j)}$$

(note that each matrix $\mathbf{\Sigma}^{(j)}$ has zeros in the columns and rows whose indices are not in $\mathcal{S}_j$). If the submatrix of $\mathbf{\Sigma}^{(j)}$ formed by taking the rows and columns with indices $\mathcal{S}_j$ is positive definite (i.e., assume the limiting conditional covariance matrix of the nonzero entries of $\boldsymbol{\xi}_k^{(j)}$ is positive definite, a reasonable assumption) and if we also assume $r_j d_j > 0$ then $\mathbf{\Sigma}$ would be a block-diagonal positive definite matrix and C6 would be satisfied. Having $r_j d_j > 0$ requires each subvector to be updated at least once per iteration and $k^{\alpha} a_k^{(j)} \to r_j > 0$.

Conditions C7 and C8. Conditions C7 and C8 are assumptions that are often applied to non-cyclic SA algorithms. C7 pertains to the uniform integrability of the squared magnitude of $\{\xi_k^{(j(z))}(\hat{\theta}_k^{(I_{z,i})})\}$ or $k^{(\beta-\alpha)/2}\xi_k^{(j(z))}(\hat{\theta}_k^{(I_{z,i})})$ (see Definition 3). A sufficient condition for C7 to hold is if the traces of the covariance matrices of the a noise terms are bounded uniformly over k. Condition C8 is impossible to verify in practice although we note that C8 is essentially identical to condition B6 of Fabian's theorem (see p. 98).

5.6 Concluding Remarks

This chapter presented a generalization of Fabian's theorem (see Theorem 2.2 in Fabian 1968 or Theorem 3 in this dissertation). The resulting generalization (Theorem 4) was used to prove the asymptotic normality of the normalized iterates form Algorithm 3 (a few additional applications of Theorem 4 are discussed in Appendix A). Attempting to use Theorem 4 to derive an asymptotic normality result for Algorithm 2 gives rise to a critical complication. First, note that when $L(\theta)$ is twice continuously differentiable then:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - \tilde{a}_k^{(j_k)} \boldsymbol{J}^{(j_k)}(\overline{\theta}_k)(\hat{\theta}_k - \theta^*) - \tilde{a}_k^{(j_k)} \beta_k^{(j_k)}(\hat{\theta}_k) - \tilde{a}_k^{(j_k)} \xi_k^{(j_k)}(\hat{\theta}_k), \qquad (5.34)$$

where $\hat{\theta}_k$ denotes an iterate from Algorithm 2 and where the ith row of $\boldsymbol{J}^{(j)}(\overline{\theta}_k)$ is equal to the ith row of $\boldsymbol{J}^{(j)}(\theta)$ evaluated at $\theta = (1-\lambda_i)\hat{\theta}_k + \lambda_i \theta^*$ for some

$\lambda_i \in [0,1]$ which depends on $\hat{\theta}_k$. Then, (5.34) may be written in the form of (5.1) by letting $\boldsymbol{W}_k = \hat{\theta}_k - \theta^*$, $\boldsymbol{\Gamma}_k = k^{\alpha}\tilde{a}_k^{(j_k)}\boldsymbol{J}^{(j_k)}(\overline{\theta}_k)$, $\boldsymbol{T}_k = -k^{\alpha+\beta/2}\tilde{a}_k^{(j_k)}\boldsymbol{\beta}_k^{(j_k)}(\hat{\theta}_k)$, $\boldsymbol{\Phi}_k = \boldsymbol{I}$, and $\boldsymbol{V}_k = -k^{(\alpha+\beta)/2}\tilde{a}_k^{(j_k)}\boldsymbol{\xi}_k^{(j_k)}(\hat{\theta}_k)$. Note, however, that $\boldsymbol{\Gamma}_k$ does not converge w.p.1 to any matrix since the random variable j_k does not converge. Therefore, Theorem 4 cannot be used to derive an asymptotic normality result for (5.34) for this definition of $\boldsymbol{\Gamma}_k$. Moreover, as discussed in pp. 103–104, it is not generally possible to find an alternate definition of $\boldsymbol{\Gamma}_k$ (along with a redefinition of $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{V}_k$) so that all of the conditions of Fabian's conditions hold without imposing more assumptions on $\hat{\theta}_k$. Future research could consider combining Algorithm 3 with iterate averaging, which may lead to the asymptotic normality of the normalized iterates.

Chapter 6

Efficiency of GCSA

This chapter attempts to quantify the relative efficiency between a strictly cyclic version of GCSA and its non-cyclic counterpart, where efficiency is defined in terms of the mean squared error (MSE) after taking into consideration the cost of implementation (the term "cost" is used in a very general sense and the precise definition is highly problem-specific). Specifically, if $\hat{\theta}_k^{\text{cyc}}$ denotes the kth iterate of a strictly cyclic implementation of GCSA and $\hat{\theta}_k^{\text{non}}$ denotes the kth iterate of its non-cyclic counterpart, the focus of this chapter is on the ratio:

$$\frac{E\|\hat{\theta}_{k_1}^{\text{cyc}} - \theta^*\|^2}{E\|\hat{\theta}_{k_2}^{\text{non}} - \theta^*\|^2}, \tag{6.1}$$

where the cumulative cost of implementing the cyclic algorithm up to iteration k_1 is equal (or approximately equal) to that of its non-cyclic counterpart up to

iteration k_2 (k_2 is then a function of k_1 and vice versa). An analogue to (6.1) would be to set the MSEs in the numerator and in the denominator to be equal and then study how k_1 and k_2 differ (e.g, Spall 1992). In general, the definition of per-iteration cost is highly problem-specific and plays an important role in computing (6.1). Section 6.1 discusses the per-iteration cost issue in more detail. Section 6.2 then obtains an asymptotic approximation to the ratio in (6.1). Section 6.3 uses the approximation from Section 6.2 to study the relative efficiency between a special case of Algorithm 3 and its non-cyclic counterpart. Lastly, Section 6.4 contains concluding remarks.

6.1 Cost of Implementation

In order to perform a fair comparison between any two algorithms it is necessary to take the cost of implementation into consideration. Here, the precise definition of "cost" varies depending on the specific algorithm at hand. For example, cost could be a measure of the number of iterations, the actual run-time of an algorithm, the number of times a certain simulation has to be run, the amount of storage required, or a combination of these. As another example, Knight et al. (2014) consider a simplified machine model with several processors and represents run-time as a linear function of the arithmetic (flop) cost and other variables (e.g., number of messages sent). Thus, if cost is defined as

run-time, the cost of implementation will be a function of the flop count as well as of other variables. This section focuses on comparing the cost of implementing the cyclic seesaw SA algorithm (the ideas can easily be generalized to the case where θ is partitioned into more than two subvectors) to the cost of implementing its non-cyclic counterpart when cost is a measure of the arithmetic computations needed to perform a single iteration. Specifically, this section pinpoints the type of arithmetic computations that can result in a significant difference between the per-iteration cost of implementing a cyclic algorithm and the per-iteration cost of implementing its non-cyclic formulation.

Letting $\hat{\theta}_k^{\mathrm{cyc}}$ denote an iterate of the cyclic seesaw SA algorithm (Section 3.1) and using the notation in (3.1) we assume throughout this section that:

$$\hat{\theta}_k^{(I)} = \begin{bmatrix} (\hat{\theta}_{k+1}^{\mathrm{cyc}})^{[\mathbf{1}]} \\ (\hat{\theta}_k^{\mathrm{cyc}})^{[\mathbf{2}]} \end{bmatrix} = \hat{\theta}_k^{\mathrm{cyc}} - \mathbf{\Phi}^{(1)}(\hat{\theta}_k^{\mathrm{cyc}}, k, \boldsymbol{V}_k) \in \mathbb{R}^p, \tag{6.2a}$$

$$\hat{\theta}_{k+1}^{\mathrm{cyc}} = \begin{bmatrix} (\hat{\theta}_{k+1}^{\mathrm{cyc}})^{[\mathbf{1}]} \\ (\hat{\theta}_{k+1}^{\mathrm{cyc}})^{[\mathbf{2}]} \end{bmatrix} = \hat{\theta}_k^{(I)} - \mathbf{\Phi}^{(2)}(\hat{\theta}_k^{(I)}, k, \boldsymbol{V}_k^{(I)}) \in \mathbb{R}^p, \tag{6.2b}$$

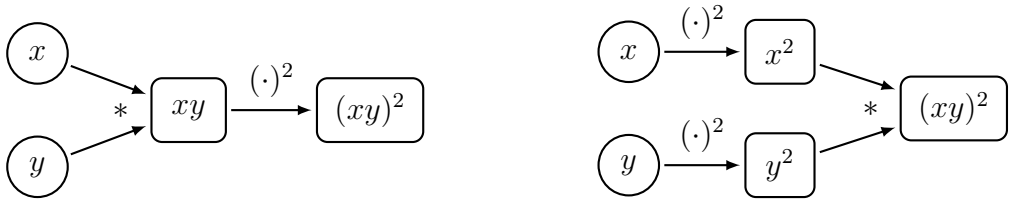
for a known vector-valued function satisfying $\mathbf{\Phi}(\theta, k, \boldsymbol{V})$: $\mathbb{R}^p \times \mathbb{Z}^+ \times \Omega_{\boldsymbol{V}} \to \mathbb{R}^p$ where $\mathbf{\Phi}^{(j)}(\cdot,\cdot,\cdot)$ is defined using the notation from (3.2) and $\boldsymbol{V}_k$ and $\boldsymbol{V}_k^{(I)}$ are random variables with values in $\Omega_{\boldsymbol{V}}$. We refer to the functions $\mathbf{\Phi}^{(1)}(\cdot,\cdot,\cdot)$ and $\mathbf{\Phi}^{(2)}(\cdot,\cdot,\cdot)$ as *update functions*. In the CSG algorithm (Section 3.2), for example, if $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{\mathrm{cyc}})$ and $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\theta}_k^{(I)})$ are obtained via the evaluation of an arithmetic func-

tion (e.g., the pure infinitesimal perturbation analysis (IPA) algorithm where the update direction is as in (2.11)) then $\mathbf{\Phi}^{(1)}(\hat{\theta}_k^{\text{cyc}}, k, \boldsymbol{V}_k) = a_k^{(1)}[\hat{\boldsymbol{g}}_k^{\text{SP}}(\hat{\theta}_k^{\text{cyc}})]^{(1)}$ and $\mathbf{\Phi}^{(2)}(\hat{\theta}_k^{(I)}, k, \boldsymbol{V}_k^{(I)}) = a_k^{(2)}[\hat{\boldsymbol{g}}^{\text{SG}}(\hat{\theta}_k^{(I)})]^{(2)}$. Similarly, if $\hat{\theta}_k^{\text{non}}$ denotes an iterate of the basic non-cyclic SA algorithm (see Section 2.2) we assume:

$$\begin{aligned}\hat{\theta}_{k+1}^{\text{non}} &= \begin{bmatrix} (\hat{\theta}_k^{\text{non}})^{[\mathbf{1}]} \\ (\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]} \end{bmatrix} \\ &= \hat{\theta}_k^{\text{non}} - \mathbf{\Phi}(\hat{\theta}_k^{\text{non}}, k, \boldsymbol{V}_k) \\ &= \hat{\theta}_k^{\text{non}} - \mathbf{\Phi}^{(1)}(\hat{\theta}_k^{\text{non}}, k, \boldsymbol{V}_k) - \mathbf{\Phi}^{(2)}(\hat{\theta}_k^{\text{non}}, k, \boldsymbol{V}_k) \in \mathbb{R}^p, \end{aligned} \tag{6.3}$$

where $\boldsymbol{V}_k$ is a random variable in $\Omega_{\boldsymbol{V}}$. Both (6.2a,b) and (6.3) may be implemented in a distributed or non-distributed manner, in the sense that the vectors $\mathbf{\Phi}^{(1)}(\cdot,\cdot,\cdot)$ and $\mathbf{\Phi}^{(2)}(\cdot,\cdot,\cdot)$ are allowed (but not required) to be computed at different locations. Botts et al. (2016), for example, consider a multi-agent surveillance and tracking problem in which agent j updates its state vector (corresponding to the vector $\theta^{[j]}$) using $\mathbf{\Phi}^{(j)}(\cdot,\cdot,\cdot)$. In this multi-agent problem, each of the vector-valued functions $\mathbf{\Phi}^{(j)}(\cdot,\cdot,\cdot)$ is computed at a different location by a different agent. This multi-agent algorithm is, therefore, a distributed algorithm. Next we define the cost of implementing (6.2a,b) and (6.3).

The definition of cost to be used in this section is based on the observation that any function, regardless of complexity, can be evaluated by perform-



(a) Requires two elementary operations. **(b)** Requires three elementary operations.

Figure 6.1: Two CGs for evaluating the function $\rho(x, y) = (xy)^2$.

ing a sequence of elementary operations involving one or two arguments at a time (e.g., addition, multiplication, division, the power operation, trigonometric functions, exponential functions, and logarithmic functions). Then, the cost of implementing an iteration of (6.3), denoted by c_k^{non}, will be defined as the number of elementary operations required to evaluate $\boldsymbol{\Phi}^{(1)}(\hat{\theta}_k^{\text{non}}, k, \boldsymbol{V}_k)$ and $\boldsymbol{\Phi}^{(2)}(\hat{\theta}_k^{\text{non}}, k, \boldsymbol{V}_k)$. Similarly, the cost of implementing an iteration of (6.2a,b), denoted by c_k^{cyc}, will be defined as the number of elementary operations required to evaluate both $\boldsymbol{\Phi}^{(1)}(\hat{\theta}_k^{\text{cyc}}, k, \boldsymbol{V}_k)$ and $\boldsymbol{\Phi}^{(2)}(\hat{\theta}_k^{(I)}, k, \boldsymbol{V}_k^{(I)})$. Note that we are omitting the cost of performing the subtraction operations appearing in (6.2a,b) and the subtraction operation appearing in (6.3). This was done due to the fact that performing these addition operations takes the same number of elementary operations, namely p, for both the cyclic and the non-cyclic algorithms. Therefore, the aforementioned subtraction operations are not significant given the focus of this section on pinpointing the sources of discrepancy between the cost of implementing a cyclic and the cost of implementing its non-cyclic counterpart.

The sequence of elementary operations performed when evaluating any

real-valued function can be represented by a computational graph (CG) (e.g., Nocedal and Wright, p. 205) which is generally not unique (see Figure 6.1). In this section we will assume that each entry of $\boldsymbol{\Phi}(\cdot,\cdot,\cdot) \in \mathbb{R}^p$ is associated with a predetermined CG whose shape is independent of the arguments of $\boldsymbol{\Phi}(\cdot,\cdot,\cdot)$. In general, the elementary computations performed when evaluating $\boldsymbol{\Phi}(\theta, k, \boldsymbol{V})$ must fall into one of the following (mutually exclusive) categories:

Cat. 1: Computations involving $\theta^{[1]}$ but not $\theta^{[2]}$.

Cat. 2: Computations involving $\theta^{[2]}$ but not $\theta^{[1]}$.

Cat. 3: Computations involving both $\theta^{[1]}$ and $\theta^{[2]}$.

Cat. 4: Computations not involving θ.

With this in mind, an elementary operation performed when computing any of the entries of $\boldsymbol{\Phi}^{(j)}(\theta, k, \boldsymbol{V})$ will be denoted by $f_\ell^{(j)}(\theta^{[1]})$, $s_\ell^{(j)}(\theta^{[1]})$, $b_\ell^{(j)}(\theta^{[1]}, \theta^{[2]})$, or $n_\ell^{(j)}(k)$ depending on whether it belongs to category 1, 2, 3, or 4 above, respectively. The letter "f" was chosen to emphasize the fact that an operation depends only on the *first* part of θ. Similarly, the letters "s", "b", and "n", were used to emphasize the fact that an operation depends on the *second* part of θ, on *both* parts of θ, or *neither* on the first- *nor* on the second part of θ. The collection of all elementary operations performed when computing the vectors $\boldsymbol{\Phi}^{(j)}(\theta, k, \boldsymbol{V})$ can then be summarized by the following sets whose elements can

be thought of as labels of the operations to be performed:

$$F^{(j)}(\theta^{[1]}) \equiv \left\{ f_\ell^{(j)}(\theta^{[1]}) \right\}_\ell, \quad S^{(j)}(\theta^{[2]}) \equiv \left\{ s_\ell^{(j)}(\theta^{[2]}) \right\}_\ell, \tag{6.4a}$$

$$B^{(j)}(\theta^{[1]}, \theta^{[2]}) \equiv \left\{ b_\ell^{(j)}(\theta^{[1]}, \theta^{[2]}) \right\}_\ell, \quad N^{(j)}(k) \equiv \left\{ n_\ell^{(j)}(k) \right\}_\ell. \tag{6.4b}$$

Next we provide insight into the reason why (6.2a,b) and (6.3) may have different per-iteration costs (i.e., why c_k^{non} may not equal c_k^{cyc}).

Consider a simple situation where $\theta = [x, y]^\top \in \mathbb{R}^2$ (so that $\theta^{[1]} = x$ and $\theta^{[2]} = y$), V is a real-valued random variable whose distribution does not depend on θ, and assume that:

$$\mathbf{\Phi}(\theta, k, V) = k^{-1} \begin{bmatrix} y(xy)^9 + V \\ 0 \end{bmatrix} + k^{-1} \begin{bmatrix} 0 \\ x(xy)^9 + V \end{bmatrix} \tag{6.5}$$

so that the first vector, respectively second vector, on the right-hand side of (6.5) represents $\mathbf{\Phi}^{(1)}(\theta, k, V)$, respectively $\mathbf{\Phi}^{(2)}(\theta, k, V)$ (in the context of Section 2.3, $k\mathbf{\Phi}(\theta, k, V)$ corresponds to the gradient of the noisy function $Q(\theta, V) = (xy)^{10}/10 + (x + y)V$). The non-cyclic algorithm in (6.3) with $\mathbf{\Phi}(\theta, k, V)$ as in (6.5) would be implemented as follows:

$$\hat{\theta}_{k+1}^{\text{non}} = \hat{\theta}_k^{\text{non}} - k^{-1} \begin{bmatrix} y_k(x_k y_k)^9 + V_k \\ 0 \end{bmatrix} - k^{-1} \begin{bmatrix} 0 \\ x_k(x_k y_k)^9 + V_k \end{bmatrix},$$

where $\hat{\theta}_k^{\text{non}} = [x_k, y_k]^\top$. Because the possibly non-zero entries of $\mathbf{\Phi}^{(1)}(\hat{\theta}_k^{\text{non}}, k, V_k)$ and $\mathbf{\Phi}^{(2)}(\hat{\theta}_k^{\text{non}}, k, V_k)$ are very similar, it may be advantageous to share certain computations in order to minimize the overall arithmetic cost. For example, the quantity $(x_k y_k)^9$ appears in both $\mathbf{\Phi}^{(1)}(\hat{\theta}_k^{\text{non}}, k, V_k)$ and $\mathbf{\Phi}^{(2)}(\hat{\theta}_k^{\text{non}}, k, V_k)$. In the multi-agent setting, if agents are allowed to communicate then the first agent could compute $(x_k y_k)^9$ and share it with the second agent. The ability to share the value of $(x_k y_k)^9$ is lost when considering a cyclic implementation (regardless of whether the implementation is distributed or not).

The implementation of the cyclic seesaw algorithm in (6.2a,b) is as follows:

$$\hat{\theta}_{k+1}^{\text{cyc}} = \hat{\theta}_k^{\text{cyc}} - k^{-1} \begin{bmatrix} y_k (x_k y_k)^9 + V_k \\ 0 \end{bmatrix} - k^{-1} \begin{bmatrix} 0 \\ x_{k+1} (x_{k+1} y_k)^9 + V_k^{(I)} \end{bmatrix},$$

where $\hat{\theta}_k^{\text{cyc}} = [x_k, y_k]^\top$ (in this case, $\hat{\theta}_k^{(I)}$ would be equal to $[x_{k+1}, y_k]^\top$). Since $(x_k y_k)^9 \neq (x_{k+1} y_k)^9$, in a cyclic implementation it would not be possible to reduce cost by sharing the value of $(xy)^9$. However, since $(xy)^9 = x^9 y^9$, a reduction in the arithmetic cost could be obtained by sharing the values of x_{k+1}^9 and y_k^9. In general, the types of arithmetic computations that can be shared (and the associated reduction in cost) depends on the CGs of the entries of $\mathbf{\Phi}(\theta, k, \boldsymbol{V})$.

In the remainder of this section we derive expressions for c_k^{non} and c_k^{cyc} that take into consideration a possible reduction in cost through the sharing of arithmetic computations. We make the following assumptions:

D0 The CG corresponding to the ith entry of $\mathbf{\Phi}(\theta, k, \boldsymbol{V})$ is the same for both the cyclic- and non-cyclic implementations and does not depend on θ, k, or $\boldsymbol{V}$.

D1 If the algorithm is implemented in a distributed manner, θ, k and $\boldsymbol{V}$ are known to the agent computing $\mathbf{\Phi}^{(j)}(\theta, k, \boldsymbol{V})$.

D2 During iteration k, it is possible to share/reuse computations corresponding to nodes in the CGs which were computed during iterations $1, \dots, k$.

Under D0–D1 let Naive-Cost$^{(1)}$ be the total number of elementary operations required to evaluate the p' CGs associated with the evaluation of $\mathbf{\Phi}^{(1)}(\cdot,\cdot,\cdot)$. Similarly, let Naive-Cost$^{(2)}$ be the total number of elementary operations required to evaluate the p' CGs associated with the evaluation of $\mathbf{\Phi}^{(2)}(\cdot,\cdot,\cdot)$. Then, under D0–D2 the following holds for both the cyclic- and non-cyclic algorithms:

$$c_k^{\text{non}} = \text{Naive-Cost}^{(1)} + \text{Naive-Cost}^{(2)} - c_{\text{share}}^{\text{non}}(k) - c_{\text{carry}}^{\text{non}}(k), \tag{6.6a}$$

$$c_k^{\text{cyc}} = \text{Naive-Cost}^{(1)} + \text{Naive-Cost}^{(2)} - c_{\text{share}}^{\text{cyc}}(k) - c_{\text{carry}}^{\text{cyc}}(k), \tag{6.6b}$$

where $c_{\text{share}}^{\text{non}}(k)$ and $c_{\text{share}}^{\text{cyc}}(k)$ denote the cost saved by sharing information on calculations performed during the kth iteration, and where $c_{\text{carry}}^{\text{non}}(k)$ and $c_{\text{carry}}^{\text{cyc}}(k)$ denote the cost saved by reusing and/or sharing computations carried from previous iterations. Next we provide expressions for $c_{\text{share}}^{\text{non}}(k)$, $c_{\text{carry}}^{\text{non}}(k)$, $c_{\text{share}}^{\text{cyc}}(k)$, and $c_{\text{carry}}^{\text{cyc}}(k)$ based on the terminology from (6.4a,b).

Let $\mathcal{A}_k$ be the set of computations performed during the kth iteration that

are shared during iteration k. Similarly, let $\mathcal{B}_k$ be the set of computations performed during iterations $1, \ldots, k-1$ that are reused and/or shared during iteration k. Additionally, let $\alpha_{F,\ell}$, $\alpha_{S,\ell}$, $\alpha_{B,\ell}$, and $\alpha_{N,\ell}$ represent the cost saved during the kth iteration by sharing $f_\ell^{(1)}(\theta^{[\mathbf{1}]})$, $s_\ell^{(1)}(\theta^{[\mathbf{2}]})$, $b_\ell^{(1)}(\theta^{[\mathbf{1}]}, \theta^{[\mathbf{2}]})$, and $n_\ell^{(1)}(k)$, respectively. Then, assuming $\alpha_{F,\ell}$, $\alpha_{S,\ell}$, $\alpha_{B,\ell}$, and $\alpha_{N,\ell}$ do not depend on θ or k:

$$
\begin{aligned}
c_{\text{share}}^{\text{non}}(k) = & \sum_\ell \alpha_{F,\ell}\, \chi\Big\{ f_\ell^{(1)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{1}]}\right) \in F^{(2)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{1}]}\right) \cap \mathcal{A}_k \Big\} \\
& + \sum_\ell \alpha_{S,\ell}\, \chi\Big\{ s_\ell^{(1)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]}\right) \in S^{(2)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]}\right) \cap \mathcal{A}_k \Big\} \\
& + \sum_\ell \alpha_{B,\ell}\, \chi\Big\{ b_\ell^{(1)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{1}]}, (\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]}\right) \in B^{(2)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{1}]}, (\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]}\right) \cap \mathcal{A}_k \Big\} \\
& + \sum_\ell \alpha_{N,\ell}\, \chi\Big\{ n_\ell^{(1)}(k) \in N^{(2)}(k) \cap \mathcal{A}_k \Big\}. \qquad (6.7)
\end{aligned}
$$

To compute $c_{\text{carry}}^{\text{non}}(k)$ let $\beta_{N,\ell}^{(j)}$ represent the arithmetic cost saved during the kth iteration by reusing the elements $n_\ell^{(j)}(i)$ for $i = 0, \ldots, k-1$. Then, assuming $\beta_{N,\ell}^{(j)}$ does not depend on θ or k:

$$
\begin{aligned}
c_{\text{carry}}^{\text{non}}(k) = & \sum_\ell \beta_{N,\ell}^{(1)}\, \chi\left\{ n_\ell^{(1)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i)\right) \cap \mathcal{B}_k \right\} \\
& + \sum_\ell \beta_{N,\ell}^{(2)}\, \chi\left\{ n_\ell^{(2)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i)\right) \cap \mathcal{B}_k \right\} \\
& - \sum_\ell \beta_{N,\ell}^{(1)}\, \chi\left\{ n_\ell^{(1)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i)\right) \cap \mathcal{B}_k \right\} \times \\
& \chi\left\{ n_\ell^{(2)} \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i)\right) \cap \mathcal{B}_k \right\}, \qquad (6.8)
\end{aligned}
$$

this follows after noting that the only terms from previous iterations that could be reused during iteration k are terms that do not depend on θ.

For the cyclic seesaw algorithm, the cost of sharing computations is different than for the non-cyclic algorithm (as was discussed on p. 135). Specifically,

$$c_{\text{share}}^{\text{cyc}}(k) = \sum_{\ell} \alpha_{S,\ell}\, \chi\Big\{ s_{\ell}^{(1)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]} \right) \in S^{(2)}\left((\hat{\theta}_k^{\text{non}})^{[\mathbf{2}]} \right) \cap \mathcal{A}_k \Big\} + \sum_{\ell} \alpha_{N,\ell}\, \chi\Big\{ n_{\ell}^{(1)}(k) \in N^{(2)}(k) \cap \mathcal{A}_k \Big\}. \tag{6.9}$$

Next, letting $\beta_{F,\ell}^{(1)}$ represent the arithmetic cost saved during the kth iteration by reusing the element $f_{\ell}^{(1)}(\hat{\theta}_k^{(1)})$ computed during the previous iteration and assuming $\beta_{F,\ell}^{(1)}$ does not depend on θ or k it follows that:

$$\begin{aligned} c_{\text{carry}}^{\text{cyc}}(k) = & \sum_{\ell} \beta_{N,\ell}^{(1)}\, \chi\left\{ n_{\ell}^{(1)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i) \right) \cap \mathcal{B}_k \right\} \\ & + \sum_{\ell} \beta_{N,\ell}^{(2)}\, \chi\left\{ n_{\ell}^{(2)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i) \right) \cap \mathcal{B}_k \right\} \\ & - \sum_{\ell} \beta_{N,\ell}^{(1)}\, \chi\left\{ n_{\ell}^{(1)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i) \right) \cap \mathcal{B}_k \right\} \times \\ & \chi\left\{ n_{\ell}^{(2)}(k) \in \bigcup_{i=0}^{k-1} \left(N^{(1)}(i) \cup N^{(2)}(i) \right) \cap \mathcal{B}_k \right\} \\ & + \sum_{\ell} \beta_{F,\ell}^{(1)}\, \chi\Big\{ f_{\ell}^{(1)}\left((\hat{\theta}_k^{\text{cyc}})^{[\mathbf{1}]} \right) \in F^{(2)}\left((\hat{\theta}_k^{\text{cyc}})^{[\mathbf{1}]} \right) \cap \mathcal{B}_k \Big\}. \end{aligned} \tag{6.10}$$

Note that (6.10) is different from (6.8) since it contains an extra term (the last line in (6.10)), which originates from the fact that the first p' entries of $\hat{\theta}_{k-1}^{(I)}$ are

the same as the first p' entries of $\hat{\theta}_k^{\text{cyc}}$. Therefore, calculations involving $(\hat{\theta}_{k-1}^{(I)})^{[1]}$ which were computed during iteration $k-1$ could be used to evaluate the computations involving $(\hat{\theta}_k^{\text{cyc}})^{[1]}$ during the kth iteration. The following Proposition compares c_k^{non} and c_k^{cyc}.

Proposition 2. Assume D0–D2 hold along with the following conditions:

D3 $\alpha_{F,\ell}$, $\alpha_{S,\ell}$, $\alpha_{B,\ell}$, and $\alpha_{N,\ell}$ do not depend on θ or k.

D4 $\beta_{N,\ell}^{(j)}$ does not depend on θ or k.

D5 $\beta_{F,\ell}^{(j)}$ does not depend on θ or k.

Then,

$$c_k^{\text{non}} - c_k^{\text{cyc}} = T_2(k) - T_1(k), \tag{6.11}$$

where $T_1(k) \geq 0$ is defined as:

$$T_1(k) \equiv \sum_{\ell} \alpha_{F,\ell}\, \chi\Big\{ f_\ell^{(1)}\left((\hat{\theta}_k^{\text{non}})^{[1]}\right) \in F^{(2)}\left((\hat{\theta}_k^{\text{non}})^{[1]}\right) \cap \mathcal{A}_k \Big\}$$
$$\sum_{\ell} \alpha_{B,\ell}\, \chi\Big\{ b_\ell^{(1)}\left((\hat{\theta}_k^{\text{non}})^{[1]}, (\hat{\theta}_k^{\text{non}})^{[2]}\right) \in B^{(2)}\left((\hat{\theta}_k^{\text{non}})^{[1]}, (\hat{\theta}_k^{\text{non}})^{[2]}\right) \cap \mathcal{A}_k \Big\}$$

and $T_2(k) \geq 0$ is defined as:

$$T_2(k) \equiv \sum_{\ell} \beta_{F,\ell}^{(1)}\, \chi\Big\{ f_\ell^{(1)}\left((\hat{\theta}_k^{\text{cyc}})^{[1]}\right) \in F^{(2)}\left((\hat{\theta}_k^{\text{cyc}})^{[1]}\right) \cap \mathcal{B}_k \Big\}.$$

Proof. First, (6.7) and (6.9) imply $T_1(k) = c^{\text{non}}_{\text{share}}(k) - c^{\text{cyc}}_{\text{share}}(k)$ while (6.8) and (6.10) imply $T_2(k) = c^{\text{cyc}}_{\text{carry}}(k) - c^{\text{non}}_{\text{carry}}(k)$. The result then follows from (6.6a,b). □

To facilitate the interpretation of (6.11) we assume the following:

D6 $\alpha^{(1)}_{F,\ell} = \beta^{(1)}_{F,\ell}$. This assumption is satisfied whenever the cost saved by sharing a computation involving only $\theta^{[\mathbf{1}]}$ is independent of whether the computation was performed during the kth iteration or during an earlier iteration.

Then, (6.11) becomes

$$
\begin{aligned}
c^{\text{non}}_k - c^{\text{cyc}}_k = &\sum_{\ell} \alpha^{(1)}_{F,\ell}\, \chi\Big\{ f^{(1)}_\ell \left((\hat{\theta}^{\text{cyc}}_k)^{[\mathbf{1}]} \right) \in F^{(2)} \left((\hat{\theta}^{\text{cyc}}_k)^{[\mathbf{1}]} \right) \cap \mathcal{B}_k \Big\} \\
&- \sum_{\ell} \alpha_{F,\ell}\, \chi\Big\{ f^{(1)}_\ell \left((\hat{\theta}^{\text{non}}_k)^{[\mathbf{1}]} \right) \in F^{(2)} \left((\hat{\theta}^{\text{non}}_k)^{[\mathbf{1}]} \right) \cap \mathcal{A}_k \Big\} \\
&- \sum_{\ell} \alpha_{B,\ell}\, \chi\Big\{ b^{(1)}_\ell \left((\hat{\theta}^{\text{non}}_k)^{[\mathbf{1}]}, (\hat{\theta}^{\text{non}}_k)^{[\mathbf{2}]} \right) \in B^{(2)} \left((\hat{\theta}^{\text{non}}_k)^{[\mathbf{1}]}, (\hat{\theta}^{\text{non}}_k)^{[\mathbf{2}]} \right) \cap \mathcal{A}_k \Big\}.
\end{aligned}
$$

Furthermore, provided it is possible to share $f^{(1)}_\ell \left(\theta^{[\mathbf{1}]} \right)$ if and only if it is possible to carry $f^{(1)}_\ell \left(\theta^{[\mathbf{1}]} \right)$ from the previous iteration, then the difference between c^{cyc}_k and c^{non}_k under D0–D6 is given by:

$$
c^{\text{cyc}}_k - c^{\text{non}}_k = \sum_{\ell} \alpha_{B,\ell}\, \chi\Big\{ b^{(1)}_\ell \left((\hat{\theta}^{\text{non}}_k)^{[\mathbf{1}]}, (\hat{\theta}^{\text{non}}_k)^{[\mathbf{2}]} \right) \in B^{(2)} \left((\hat{\theta}^{\text{non}}_k)^{[\mathbf{1}]}, (\hat{\theta}^{\text{non}}_k)^{[\mathbf{2}]} \right) \cap \mathcal{A}_k \Big\}.
$$

Here, we can see that under conditions D0–D6 the main difference between the arithmetic cost of a cyclic algorithm and that of its non-cyclic counterpart

is due to sharing elements that combine information regarding both subvectors of θ. Intuitively, this is due to the fact that by implementing an algorithm in a cyclic manner we lose the ability to reuse any computations involving both $\theta^{[1]}$ and $\theta^{[2]}$. Therefore, if a significant cost reduction is achieved by sharing this type of element then an iteration of the non-cyclic algorithm will have a significantly lower cost.

In practice, the per-iteration cost of implementation will depend strongly on the specific algorithm considered. For the cyclic seesaw SPSA algorithm, for example, a more appropriate definition of cost might be the number of noisy loss function evaluations used per-iteration. If cyclic seesaw SPSA is implemented according to (3.7) along with (3.10) and (3.11) then the per-iteration cost of implementation would be equal to 4 loss function measurements per-iteration. In the case of the regular SPSA algorithm (see (2.13)), however, the per-iteration cost of implementation would be only 2 loss function measurements per-iteration. The following section presents some general results on the relative asymptotic efficiency in (6.1).

6.2 Approximating Relative Efficiency

In the previous chapter (see Section 5.4) it was shown that (under certain assumptions) $k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*) \xrightarrow{\text{dist}} \mathcal{N}(\mu^{\text{cyc}}, \Sigma^{\text{cyc}})$ for some mean vector μ^{cyc}

and covariance matrix Σ^{cyc}, where $\hat{\theta}_k^{\text{cyc}}$ denotes the kth iterate of Algorithm 3 and the notation "$\xrightarrow{\text{dist}}$" means convergence in distribution. A similar result is readily obtained for its non-cyclic counterpart. In particular, let $\hat{\theta}_k^{\text{non}}$ denote the kth iterate obtained using the non-cyclic counterpart of Algorithm 3. Then, under certain assumptions (see Fabian 1968) it can be shown that $k^{\beta/2}(\hat{\theta}_k^{\text{non}} - \theta^*) \xrightarrow{\text{dist}} \mathcal{N}(\mu^{\text{non}}, \Sigma^{\text{non}})$ for some mean vector μ^{non} and covariance matrix Σ^{non}. This section shows how results on asymptotic normality can be used to approximate the relative efficiency in (6.1).

Following the ideas in (Spall, 1992) we assume:

$$\lim_{k\to\infty} E[k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)] = \mu^{\text{cyc}}, \quad \lim_{k\to\infty} E[k^{\beta/2}(\hat{\theta}_k^{\text{non}} - \theta^*)] = \mu^{\text{non}}, \tag{6.12}$$

and that:

$$\lim_{k\to\infty} \text{Var}\,[k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)] = \Sigma^{\text{cyc}}, \quad \lim_{k\to\infty} \text{Var}\,[k^{\beta/2}(\hat{\theta}_k^{\text{non}} - \theta^*)] = \Sigma^{\text{non}}. \tag{6.13}$$

Spall (1992) discusses how (6.12) and (6.13) hold if the terms $\|k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)\|^2$ and $\|k^{\beta/2}(\hat{\theta}_k^{\text{non}} - \theta^*)\|^2$ are uniformly integrable (see also Laha and Rohatgi, pp. 138–140). Equations (6.12) and (6.13) allow us to obtain the following approximation for large k:

$$E\|\hat{\theta}_k^{\text{cyc}} - \theta^*\|^2 = \text{trace}\left[E(\hat{\theta}_k^{\text{cyc}} - \theta^*)(\hat{\theta}_k^{\text{cyc}} - \theta^*)^\top\right] \approx k^{-\beta}\,\text{trace}\left[\Sigma^{\text{cyc}} + \mu^{\text{cyc}}(\mu^{\text{cyc}})^\top\right].$$

Similarly, for large k we approximate:

$$E\|\hat{\theta}_k^{\text{non}} - \theta^*\|^2 \approx k^{-\beta} \operatorname{trace}\left[\Sigma^{\text{non}} + \mu^{\text{non}}(\mu^{\text{non}})^\top\right].$$

Then, for large k_1 and k_2 we may approximate (6.1) as follows:

$$\frac{E\|\hat{\theta}_{k_1}^{\text{cyc}} - \theta^*\|^2}{E\|\hat{\theta}_{k_2}^{\text{non}} - \theta^*\|^2} \approx \left(\frac{k_2}{k_1}\right)^{\beta} \frac{\operatorname{trace}\left[\Sigma^{\text{cyc}}\right] + \left[\mu^{\text{cyc}}\right]^\top \mu^{\text{cyc}}}{\operatorname{trace}\left[\Sigma^{\text{non}}\right] + \left[\mu^{\text{non}}\right]^\top \mu^{\text{non}}}. \tag{6.14}$$

Here, we remind the reader that in order to perform a fair comparison between the cyclic and non-cyclic algorithm, the cumulative cost of implementing the cyclic algorithm up to iteration k_1 is assumed to be equal (or approximately equal) to that of its non-cyclic counterpart up to iteration k_2; an analysis of the asymptotic properties of (6.14) would therefore require increasing k_1 and k_2 at a rate that ensures the comparison is fair at all times. With this in mind, we introduce the following proposition.

Proposition 3. Let $0 < c^{\text{cyc}}$ and $0 < c^{\text{non}}$ be finite constants such that $c^{\text{cyc}}/c^{\text{non}}$ is a rational number. Then, there exists an infinite sequence $\{(k_{1(i)}, k_{2(i)})\}_{i=1}^{\infty}$ in $\mathbb{Z}^+ \times \mathbb{Z}^+$ with $k_{1(i-1)} < k_{1(i)}$ and $k_{2(i-1)} < k_{2(i)}$ such that $c^{\text{cyc}} k_{1(i)} = c^{\text{non}} k_{2(i)}$.

Proof. Let $c^{\text{cyc}}/c^{\text{non}} = p/q$ where p and q are strictly positive integers. The desired sequence can be obtained by letting $k_{1(i)} = qi$ and $k_{2(i)} = pi$. □

The significance of Proposition 3 is that if the per-iteration costs c_k^{cyc} and c_k^{non}

are nonzero, finite, independent of k (so that $c_k^{\text{cyc}} = c^{\text{cyc}}$ and $c_k^{\text{non}} = c^{\text{non}}$), and if $c^{\text{cyc}}/c^{\text{non}}$ is a rational number, then the cost of computing $\hat{\theta}_{k_{1(i)}}^{\text{cyc}}$ is exactly equal to the cost of computing $\hat{\theta}_{k_{2(i)}}^{\text{non}}$ for any strictly-positive integer i when $k_{1(i)}$ and $k_{2(i)}$ are as in Proposition 3. We can then obtain the following result:

$$\lim_{i\to\infty} \frac{E\|\hat{\theta}_{k_{1(i)}}^{\text{cyc}} - \theta^*\|^2}{E\|\hat{\theta}_{k_{2(i)}}^{\text{non}} - \theta^*\|^2} = \left(\frac{c^{\text{cyc}}}{c^{\text{non}}}\right)^{\beta} \frac{\text{trace}\left[\Sigma^{\text{cyc}}\right] + \left[\mu^{\text{cyc}}\right]^\top \mu^{\text{cyc}}}{\text{trace}\left[\Sigma^{\text{non}}\right] + \left[\mu^{\text{non}}\right]^\top \mu^{\text{non}}}. \tag{6.15}$$

The following section uses the limit in (6.15) to study the relative efficiency between a special case of Algorithm 3 and its non-cyclic counterpart.

6.3 Relative Efficiency: A Special Case

This section computes the asymptotic relative efficiency between a non-cyclic algorithm and a cyclic algorithm which is a special case of Algorithm 3 in which the sets $\mathcal{S}_j$ do not intersect (this is equivalent to saying that any single entry of $\hat{\theta}_k$ can only be updated by one processor/agent although each processor/agent may update several entries) and updates are produced in a strictly cyclic manner; this cyclic algorithm is described in more detail in Algorithm 4. Given our desire to analyze the effect of a cyclic implementation on asymptotic efficiency, an appropriate non-cyclic counterpart to Algorithm 4 is given in 5. Here, in order to to isolate the effects of a cyclic implementation on asymptotic efficiency, a diagonal matrix of gains was used in an attempt to make the non-

Algorithm 4 Strictly Cyclic Version of Algorithm 3

Require: $\hat{\theta}_0^{\text{cyc}}$, $\{a_k^{(j)}\}_{k\geq 0}$ for $j = 1, \ldots, d$, and let $\mathcal{S}_1, \ldots, \mathcal{S}_d$ be such that (4.36) holds. Set $k = 0$.

1: **while** stopping criterion has not been reached **do**
2: Set $\hat{\theta}_k^{(I_0)} = \hat{\theta}_k^{\text{cyc}}$.
3: **for** $j = 1, \ldots, d$ **do**
4: Define $\hat{\theta}_k^{(I_j)} \equiv \hat{\theta}_k^{(I_{j-1})} - a_k^{(j)}\, \hat{\boldsymbol{g}}^{(j)}\left(\hat{\theta}_k^{(I_{j-1})}\right)$.
5: **end for**
6: Let $\hat{\theta}_{k+1}^{\text{cyc}} \equiv \hat{\theta}_k^{(I_d)}$.
7: set $k = k + 1$
8: **end while**

cyclic algorithm as similar to the cyclic algorithm as possible. In the remainder of this section we obtain expressions for the terms Σ^{cyc}, Σ^{non}, μ^{cyc}, and μ^{non} (defined in the opening paragraph of Section 6.2) in order to estimate the relative efficiency between Algorithms 4 and 5 in the manner of (6.15).

The following corollary to Theorem 5 will be used to derive expressions for Σ^{cyc} and μ^{cyc}, the parameters of the asymptotic distribution of $k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$.

Corollary 4. Let $\hat{\theta}_k^{\text{cyc}}$ be generated according to Algorithm 4. Additionally, let $\mathcal{F}_k$ denote the sigma field generated by $\{\hat{\theta}_\ell^{\text{cyc}}\}_{\ell=0}^k$ as well as by any random variables generated by the algorithm in the production of $\hat{\theta}_k^{\text{cyc}}$. Also, assume the following conditions hold (all scalars, matrices, and vectors are assumed to have real entries):

C0′ $L(\theta)$ is twice continuously differentiable, $\boldsymbol{H}(\theta)$ has bounded entries, and

Algorithm 5 Non-Cyclic Counterpart to Algorithm 4

Require: $\hat{\theta}_0^{\text{non}}$, $\{a_k^{(j)}\}_{k\geq 0}$ for $j = 1, \ldots, d$, let $\mathcal{S}_1, \ldots, \mathcal{S}_d$ be such that (4.36) holds, and let $\boldsymbol{A}_k$ be a diagonal matrix with ith diagonal entry equal to $a_k^{(j)}$, where j is such that $i \in \mathcal{S}_j$. Set $k = 0$.

1: **while** stopping criterion has not been reached **do**

2: Let $\hat{\theta}_{k+1}^{\text{non}} = \hat{\theta}_k^{\text{non}} - \boldsymbol{A}_k \hat{\boldsymbol{g}}(\hat{\theta}_k^{\text{non}})$.

3: set $k = k + 1$

4: **end while**

$\hat{\theta}_k^{(I_j)} \to \theta^*$ w.p.1 for $j = 1, \ldots, d$.

C1′ The same as C1.

C2′ There exists an diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{p\times p}$ with strictly positive eigenvalues and a nonsingular matrix $\boldsymbol{S} \in \mathbb{R}^{p\times p}$ such that $\boldsymbol{\Gamma} \equiv \sum_{j=1}^{d} r_j \boldsymbol{J}^{(j)}(\theta^*) = \boldsymbol{A}\boldsymbol{H}(\theta^*) = \boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{S}^{-1}$ where $\boldsymbol{A}$ is the diagonal matrix with ith diagonal entry equal to r_i, and where $\boldsymbol{J}^{(j)}(\theta)$ denotes the Jacobian of $\boldsymbol{g}^{(j)}(\theta)$.

C3′ β and α satisfy $0 < \beta \leq \alpha$. Additionally, for all j there exist a finite vector $\boldsymbol{b}^{(j)} \in \mathbb{R}^p$ such that $k^{\beta/2}\boldsymbol{\beta}^{(j)}(\hat{\theta}_k^{(I_j)}) \to \boldsymbol{b}^{(j)}$ w.p.1 for all j.

C4′ For all j, $E[\boldsymbol{\xi}^{(j)}(\hat{\theta}_k^{(I_j)})|\hat{\theta}_k^{(I_j)}] = \boldsymbol{0}$.

C5′ There exists a constant $C > 0$ and a matrix $\boldsymbol{\Sigma}^{(j)}$ such that one of the following holds w.p.1:

(i) $\beta = \alpha$, $C > \|E[\boldsymbol{\xi}^{(j)}\big(\hat{\theta}_k^{(I_j)}\big)[\boldsymbol{\xi}^{(j)}\big(\hat{\theta}_k^{(I_j)}\big)]^\top|\mathcal{F}_k] - \boldsymbol{\Sigma}^{(j)}\| \to 0$.

(ii) $\beta < \alpha$, $C > \|k^{\beta-\alpha}E[\boldsymbol{\xi}^{(j)}\big(\hat{\theta}_k^{(I_j)}\big)[\boldsymbol{\xi}^{(j)}\big(\hat{\theta}_k^{(I_j)}\big)]^\top|\mathcal{F}_k] - \boldsymbol{\Sigma}^{(j)}\| \to 0$.

Additionally, if $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two different elements of $\{\boldsymbol{\xi}_k^{(j)}(\hat{\theta}_k^{(I_j)})\}_j$, then one of the following holds w.p.1:

(iii) C5′-(i) holds and $C > \|E[\boldsymbol{v}_1\boldsymbol{v}_2^\top|\mathcal{F}_k]\| \to 0$.

(iv) C5′-(ii) holds and $C > \|k^{\beta-\alpha}E[\boldsymbol{v}_1\boldsymbol{v}_2^\top|\mathcal{F}_k]\| \to 0$.

C6′ Let $\boldsymbol{M}_B \equiv \sum_{j=1}^d \boldsymbol{\Sigma}^{(j)}$. The matrix $\boldsymbol{A}\boldsymbol{M}_B\boldsymbol{A}$ is positive definite.

C7′ One of the following holds:

(i) $\beta = \alpha$ and B5-(i) holds with $\boldsymbol{V}_k$ replaced by $\boldsymbol{\xi}^{(j)}(\hat{\theta}_k^{(I_j)})$.

(ii) $\beta < \alpha$ and B5-(i) holds with $\boldsymbol{V}_k$ replaced by $k^{(\beta-\alpha)/2}\boldsymbol{\xi}^{(j)}(\hat{\theta}_k^{(I_j)})$.

C8′ The same as B6′ with U_{ii} replaced by Λ_{ii}.

Then, $k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*) \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu}^{\text{cyc}}, \boldsymbol{\Sigma}^{\text{cyc}})$ with $\boldsymbol{\mu}^{\text{cyc}} = -\boldsymbol{S}[\boldsymbol{\Lambda} - \beta_+/2\boldsymbol{I}]^{-1}\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{b}^{\text{cyc}}$ and $\boldsymbol{\Sigma}^{\text{cyc}} = \boldsymbol{S}\boldsymbol{Q}^{\text{cyc}}\boldsymbol{S}^\top$, where

$$\boldsymbol{b}^{\text{cyc}} \equiv \sum_{j=1}^d \boldsymbol{b}^{(j)}, \quad Q_{ij}^{\text{cyc}} = \frac{\left[\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{M}_B\boldsymbol{A}(\boldsymbol{S}^{-1})^\top\right]_{ij}}{\Lambda_{ii} + \Lambda_{jj} - \beta_+}. \tag{6.16}$$

Proof. Since C0′–C8′ imply C0–C8, the result follows from Theorem 5. □

The following corollary to Theorem 5 will be used to derive expressions for $\boldsymbol{\Sigma}^{\text{non}}$ and $\boldsymbol{\mu}^{\text{non}}$, the parameters of the asymptotic distribution of $k^{\beta/2}(\hat{\theta}_k^{\text{non}} - \theta^*)$.

Corollary 5. Let $\hat{\theta}_k^{\text{non}}$ be generated according to Algorithm 5. Additionally, let $\mathcal{F}_k$ denote the sigma field generated by $\{\hat{\theta}_\ell^{\text{non}}\}_{\ell=0}^k$ as well as by any random

variables generated by the algorithm in the production of $\hat{\theta}_k^{\text{non}}$. Also, assume the following conditions hold (all scalars, matrices, and vectors are assumed to have real entries):

C0″ $L(\theta)$ is twice continuously differentiable, $\boldsymbol{H}(\theta)$ has bounded entries, and $\hat{\theta}_k^{\text{non}} \to \theta^*$ w.p.1.

C3″ β and α satisfy $0 < \beta \leq \alpha$. Additionally, there exist a finite vector $\boldsymbol{b}^{\text{non}} \in \mathbb{R}^p$ such that $k^{\beta/2}\beta\big(\hat{\theta}_k\big) \to \boldsymbol{b}^{\text{non}}$ w.p.1.

C4″ $E[\xi\big(\hat{\theta}_k\big)|\hat{\theta}_k] = 0$.

C5″ There exists a constant $C > 0$ and a matrix $\boldsymbol{M}$ such that one of the following holds w.p.1:

(i) $\beta = \alpha$, $C > \|E[\xi\big(\hat{\theta}_k\big)[\xi\big(\hat{\theta}_k\big)]^\top|\mathcal{F}_k] - \boldsymbol{M}\| \to 0$.

(ii) $\beta < \alpha$, $C > \|k^{\beta-\alpha}E[\xi\big(\hat{\theta}_k\big)[\xi\big(\hat{\theta}_k\big)]^\top|\mathcal{F}_k] - \boldsymbol{M}\| \to 0$.

C6″ $\Sigma \equiv \boldsymbol{A}\boldsymbol{M}\boldsymbol{A}$ is a positive definite matrix.

C7″ One of the following holds:

(i) $\beta = \alpha$ and B5-(i) holds with $\boldsymbol{V}_k$ replaced by $\xi\big(\hat{\theta}_k\big)$.

(ii) $\beta < \alpha$ and B5-(i) holds with $\boldsymbol{V}_k$ replaced by $k^{(\beta-\alpha)/2}\xi\big(\hat{\theta}_k\big)$.

C1″, C2″, & C8″ The same as C1′, C2′, and C8′, respectively.

Then, $k^{\beta/2}(\hat{\theta}_k^{\text{non}} - \theta^*) \xrightarrow{\text{dist}} \mathcal{N}(\mu^{\text{non}}, \Sigma^{\text{non}})$ with $\mu^{\text{non}} = -\boldsymbol{S}[\boldsymbol{\Lambda} - \beta_+/2\boldsymbol{I}]^{-1}\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{b}^{\text{non}}$ and $\Sigma^{\text{non}} = \boldsymbol{S}\boldsymbol{Q}^{\text{non}}\boldsymbol{S}^\top$, where

$$Q_{ij}^{\text{non}} = \frac{\left[\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{M}\boldsymbol{A}(\boldsymbol{S}^{-1})^\top\right]_{ij}}{\Lambda_{ii} + \Lambda_{jj} - \beta_+}. \tag{6.17}$$

Proof. Since C0″–C8″ imply C0–C8, the result follows from Theorem 5 □

The validity of C0′–C8′ and C0″–C8″ is directly related to the validity of C0–C8 (see Section 5.5 for an in-depth discussion on this topic). Using the expressions for Σ^{cyc}, μ^{cyc}, Σ^{non}, and μ^{non} from Corollaries 4 and 5 along with the values of c^{non} and c^{cyc} we can compute the estimate in (6.15). In general, however, the study of this ratio is rather complicated. In order to obtain a simple expression for (6.15) we will consider a special case.

Proposition 4. Let $\hat{\theta}_k^{\text{cyc}}$ be obtained according to Algorithm 4 and let $\hat{\theta}_k^{\text{non}}$ be obtained according to Algorithm 5. Assume the following:

E0 Conditions C0′–C8′ and C0″–C8″ hold.

E1 $\beta > 0$ and $a_k^{(j)} > 0$ for $j = 1, \dots, d$ are the same for both Algorithms 4 and 5.

E2 $\boldsymbol{b}^{\text{cyc}} = \boldsymbol{b}^{\text{non}}$ (e.g., as in the SG and CSG algorithms).

E3 At least one of the following holds:

(i) $\boldsymbol{S} = c\boldsymbol{I}$ for some constant c (i.e., $\boldsymbol{H}(\theta^*)$ is a diagonal matrix).

(ii) $\boldsymbol{M}_B = \boldsymbol{M}$ (e.g., when the entries of the $\boldsymbol{\xi}_k$ are independent given $\mathcal{F}_k$).

Then, when $k_{1(i)}$ and $k_{2(i)}$ are as in Proposition 3 the limit in (6.15) becomes:

$$\lim_{i\to\infty} \frac{E\|\hat{\boldsymbol{\theta}}^{\text{cyc}}_{k_{1(i)}} - \boldsymbol{\theta}^*\|^2}{E\|\hat{\boldsymbol{\theta}}^{\text{non}}_{k_{2(i)}} - \boldsymbol{\theta}^*\|^2} = \left(\frac{c^{\text{cyc}}}{c^{\text{non}}}\right)^{\beta}. \tag{6.18}$$

Proof. First, note that E0 implies the results of Corollaries 4 and 5 hold. Then, E1 and E2 imply $\boldsymbol{\mu}^{\text{cyc}} = \boldsymbol{\mu}^{\text{non}}$. Next, E2 along with (6.16) and (6.17) implies that $\text{trace}\,[\boldsymbol{\Sigma}^{\text{cyc}}] = \text{trace}\,[\boldsymbol{\Sigma}^{\text{non}}]$. The relationship in (6.18) then follows from (6.15). □

An implication of Proposition 4 is that the asymptotic relative efficiency is entirely determined by c^{cyc} and c^{non}. If $c^{\text{cyc}}/c^{\text{non}} > 1$ then the non-cyclic algorithm is more efficient, if $c^{\text{cyc}}/c^{\text{non}} < 1$ then the cyclic algorithm is more efficient, and if $c^{\text{cyc}}/c^{\text{non}} = 1$ then both algorithms are equally efficient. In general, however, it is not easy to see whether ratio in (6.15) is greater than one, less than one, or equal to one. Next we show that when E3 does not hold then having $c^{\text{cyc}}/c^{\text{non}} < 1$ does not imply the cyclic algorithm is asymptotically more efficient. Similarly, having $c^{\text{cyc}}/c^{\text{non}} > 1$ does not imply the non-cyclic algorithm is asymptotically more efficient and having $c^{\text{cyc}}/c^{\text{non}} = 1$ does not imply both algorithms are asymptotically equally efficient.

Let us assume that condition E3 does not hold. Specifically, the matrix $\boldsymbol{S}$ is not a multiple of the identity matrix and that $\boldsymbol{M}_B \neq \boldsymbol{M}$. In this case, the limit of the ratio in (6.15) may be greater than one even when $c^{\text{cyc}}/c^{\text{non}} < 1$, less

than one even when $c^{\text{cyc}}/c^{\text{non}} > 1$, and different from one even when $c^{\text{cyc}}/c^{\text{non}} = 1$. To illustrate this fact we consider the special case where $\boldsymbol{b}^{\text{cyc}}$ and $\boldsymbol{b}^{\text{non}}$, the asymptotic values of normalized bias vectors (see (6.16) and conditions C3′ and C3″ for the formal definitions of these vectors), are identically 0. This is a valid assumption when the update directions are SG-based. Here,

$$\lim_{i\to\infty} \frac{E\|\hat{\theta}^{\text{cyc}}_{k_{1(i)}} - \theta^*\|^2}{E\|\hat{\theta}^{\text{non}}_{k_{2(i)}} - \theta^*\|^2} = \left(\frac{c^{\text{cyc}}}{c^{\text{non}}}\right)^{\beta} \frac{\operatorname{trace}\left[\boldsymbol{\Sigma}^{\text{cyc}}\right]}{\operatorname{trace}\left[\boldsymbol{\Sigma}^{\text{non}}\right]}. \tag{6.19}$$

Furthermore, we assume $\boldsymbol{A} = \boldsymbol{I}$, $\alpha < 1$ (so that $\beta_+ = 0$ in condition B6′) and that the block-diagonal matrix $\boldsymbol{M}_B$ (appearing in condition C5′) coincides with $\boldsymbol{M}$ (appearing in condition C5″) on the block-diagonal part of $\boldsymbol{M}_B$. Specifically, there exist block matrices $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_p$ such that:

$$\boldsymbol{M}_B = \begin{bmatrix} \boldsymbol{B}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{B}_d \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{B}_1 & * & * \\ * & \ddots & * \\ * & * & \boldsymbol{B}_d \end{bmatrix}, \tag{6.20}$$

where $*$ denote possibly non-zero entries and $\mathbf{0}$ denotes a matrix of zeros of the appropriate dimension. Let us discuss a situation where the assumption in (6.20) is satisfied. First, under conditions C4′ and C4″ it follows from (5.30) that $\boldsymbol{M}_B$ corresponds to the limiting (conditional) covariance of $k^{(\beta-\alpha)/2}[\xi_k^{(1)}(\hat{\theta}_k^{(I_1)}) + \cdots + \xi_k^{(d)}(\hat{\theta}_k^{(I_d)})]$ while $\boldsymbol{M}$ represents the limiting (conditional) covariance of $k^{(\beta-\alpha)/2}\xi_k(\hat{\theta}_k^{\text{non}})$. Under condition C5′-(iii) and C5′-(iv), however, it follows that

$\boldsymbol{M}_B$ must be a block diagonal matrix (i.e., $\boldsymbol{M}_B$ must have the form in (6.20)), this contrasts with the matrix $\boldsymbol{M}$ which may have extra nonzero entries due to the possible asymptotic correlation between the entries of ξ_k.

Consider now the following example. Let :

$$\boldsymbol{H}(\theta^*) = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \boldsymbol{M}_B = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}. \tag{6.21}$$

The positive definite matrix $\boldsymbol{H}(\theta^*)$ (with eigenvalues 1 and 3) might correspond, for example, to the loss function $L(\theta) = \theta^\top \boldsymbol{H}(\theta^*)\theta$. Then $\Gamma = \boldsymbol{A}\boldsymbol{H}(\theta^*) = \boldsymbol{H}(\theta^*)$ is positive definite and satisfies:

$$\Gamma = \boldsymbol{S} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \boldsymbol{S}^{-1}, \text{ with } \boldsymbol{S} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Under the above conditions and using the definitions of Σ^{cyc} and Σ^{non} given in Corollaries 4 and 5: $\operatorname{trace}[\Sigma^{\text{cyc}}]/\operatorname{trace}[\Sigma^{\text{non}}] = 4/5$. Here, (6.19) implies that the *cyclic algorithm is asymptotically more efficient if and only if* $c^{\text{cyc}}/c^{\text{non}} < (5/4)^{-\beta}$. Now, say we modify $\boldsymbol{M}_B$ and $\boldsymbol{M}$ as follows (while keeping $\boldsymbol{H}(\theta^*)$ as in (6.21)):

$$\boldsymbol{M}_B = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

Then, $\operatorname{trace}[\Sigma^{\text{cyc}}]/\operatorname{trace}[\Sigma^{\text{non}}] = 4/3$. In this case, (6.19) implies that the *cyclic algorithm is asymptotically more efficient if and only if* $c^{\text{cyc}}/c^{\text{non}} < (3/4)^{-\beta}$. This

example illustrates the fact that there is a delicate balance between $\boldsymbol{S}$, $\boldsymbol{\Lambda}$, $\boldsymbol{A}$, $\boldsymbol{M}$, β, and $c^{\text{cyc}}/c^{\text{non}}$ that determines which algorithm is more efficient.

6.4 Concluding Remarks

In order to perform a fair comparison between any two algorithms it is necessary to define the cost of implementing each algorithm. Generally, the definition of "cost" is highly problem-specific. This chapter begins by considering the case where cost is a measure of the number of arithmetic operations needed to perform a single iteration. Under certain assumptions, Section 6.1 pinpointed the type of arithmetic operations that result in a difference between the cost of implementing the cyclic seesaw algorithm and the cost of implementing its non-cyclic counterpart. This chapter also provided an approximation to the limiting value of the relative efficiency in (6.1), which takes into consideration the per-iteration costs of implementing the cyclic- and non-cyclic algorithms, for comparing Algorithm 4 to Algorithm 5. The approximation to the asymptotic relative efficiency was based on the asymptotic normality results from Section 5.4. This chapter also showed that the approximate asymptotic relative efficiency is, under certain conditions, entirely determined by the per-iteration costs of implementing the two algorithms. In general, however, there is a delicate balance between the gain sequences, the covariance matrix of the noise

terms, the Hessian matrix of $L(\theta)$, and the per-iteration implementation costs that determines which method is more efficient.

Chapter 7

Numerical Analysis

Chapter 4 of this dissertation derived conditions for the convergence of the GCSA algorithm, Chapter 5 derived conditions for the asymptotic normality of a special case of GCSA, and Chapter 6 discussed the asymptotic relative efficiency between a special case of GCSA and its non-cyclic counterpart. This chapter contains numerical results that illustrate the theory of Chapters 4–6. Specifically, Section 7.1 contains numerical results on the convergence of the GCSA algorithm, Section 7.2 contains results on asymptotic normality, and Section 7.3 contains results on relative efficiency.

7.1 Numerical Examples on Convergence

This section provides a few numerical examples on the convergence of Algorithms 2 (where the subvector to update is selected at random) and 3 (where

the subvector to update is selected following a deterministic pattern). Each of these two algorithms is implemented using SPSA-based noisy update directions (Section 7.1.1) and SG-based noisy update directions (Section 7.1.2). Algorithms 2 and 3 are also compared to two non-cyclic variants.

7.1.1 Algorithms 2 & 3 with SPSA-Based Gradient Estimates

This section considers the problem of stochastic optimization using Algorithms 2 and 3 (special cases of GCSA) with SPSA-based gradient estimates of the form in (3.10) and (3.11). Here only the subvector to update is perturbed at each subiteration. The loss function to minimize is the *skewed quartic* function (Spall 2003, Chapter 6):

$$L(\theta) = (\boldsymbol{B}\theta)^\top \boldsymbol{B}\theta + 0.1\sum_{i=1}^{p}(\boldsymbol{B}\theta)_i^3 + 0.01\sum_{i=1}^{p}(\boldsymbol{B}\theta)_i^4, \tag{7.1}$$

where $p = 10$ is the dimension of θ, $p\boldsymbol{B} \in \mathbb{R}^{p\times p}$ is an upper-triangular matrix of ones, and $\boldsymbol{v}_i$ denotes the ith entry of the vector $\boldsymbol{v}$. $L(\theta)$ achieves its minimum at $\theta^* = 0$ with $L(\theta^*) = 0$. Figure 7.1 illustrates the function in (7.1) for the special case where $p = 2$. We assume that the loss function is corrupted by i.i.d. Gaussian noise, that is $Q(\theta, \boldsymbol{V}) = L(\theta) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 0.1^2)$.

In order to implement Algorithms 2 and 3 the vector $\theta = [\tau_1, \ldots, \tau_{10}]^\top$ is divided into $d = 5$ subvectors of length 2 where the jth subvector is equal to $[\tau_{2j-1}, \tau_{2j}]^\top$. The values for $a_k^{(j)}$ and $c_k^{(j)}$ are given by $a_k^{(j)} = a/(1+k)^\alpha$ and

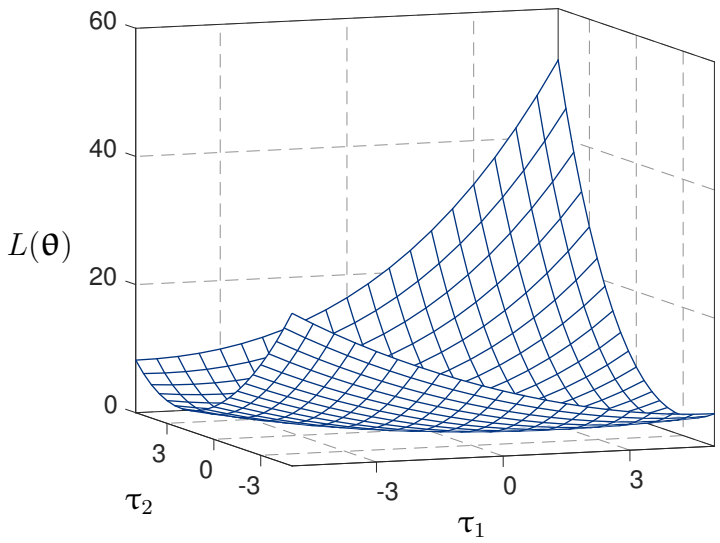


Figure 7.1: The skewed quartic function in two dimensions. In this example, $\theta = [\tau_1, \tau_2]^\top \in \mathbb{R}^2$ and $L(\theta)$ is defined as in (7.1).

$c_k^{(j)} = c/(1+k)^\gamma$ for all j with $a = 1$, $\alpha = 0.602$, $c = 1$, and $\gamma = 0.101$. These values of α and γ are standard values for the SPSA algorithm while the values of a, A, and c were *lightly* tuned. The non-zero entries of the perturbation vectors (the non-zero entries of $\boldsymbol{\Delta}$) are independent and take the value ± 1 with equal probability. For Algorithm 2, all subvectors were equally likely to be selected for updating (i.e., $q(j) = 1/5$ for all j). We let $\hat{\theta}_u^{\text{Alg 2}}$ and $\hat{\theta}_u^{\text{Alg 3}}$ denote the iterates of Algorithms 2 and 3, respectively, after having performed u subvector updates (in general, the number of subvector updates is not necessarily equal to the number of iterations). Figure 7.2 presents the values of $\overline{\|\hat{\theta}_u^{\text{Alg 2}} - \theta^*\|}/\|\hat{\theta}_0^{\text{Alg 2}} - \theta^*\|$ and $\overline{\|\hat{\theta}_u^{\text{Alg 3}} - \theta^*\|}/\|\hat{\theta}_0^{\text{Alg 3}} - \theta^*\|$ with $\hat{\theta}_0 \equiv \hat{\theta}_0^{\text{Alg 2}} = \hat{\theta}_0^{\text{Alg 3}} = [1, \ldots, 1]^\top$, where $\bar{\theta}_u^{\text{Alg 2}}$ and $\bar{\theta}_u^{\text{Alg 3}}$ denote the mean values of the algorithm iterates over 100 replications. It is seen that the mean distance between $\hat{\theta}_u^{\text{Alg 2}}$ and θ^* (rela-

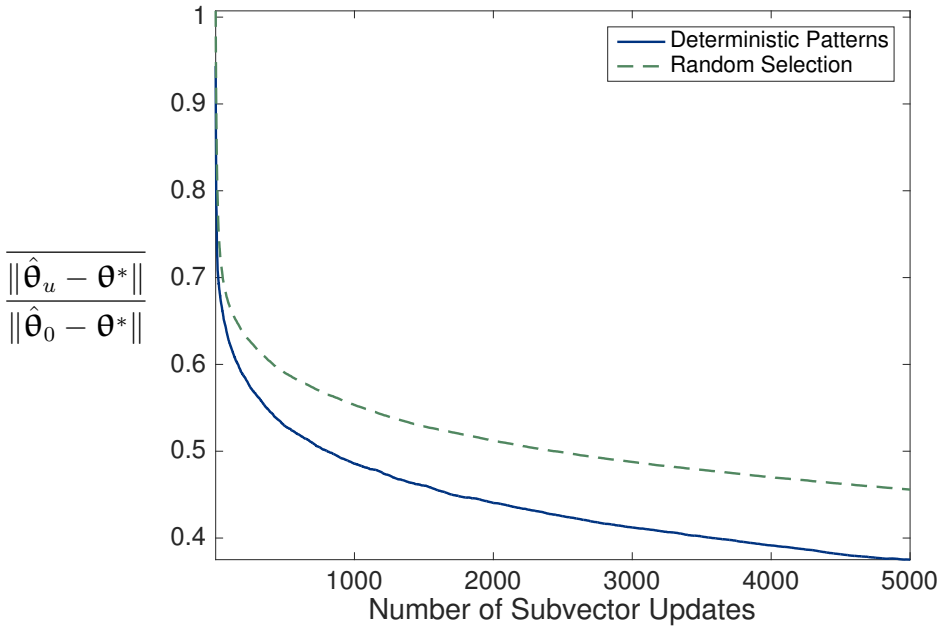


Figure 7.2: An illustration of Algorithms 2 (randomized selection) and 3 (deterministic patterns) on the noisy skewed quartic loss function when the gradient estimates are SPSA-based. Here, $\hat{\theta}_u$ represents either $\hat{\theta}_u^{\text{Alg 2}}$ or $\hat{\theta}_u^{\text{Alg 3}}$.

tive to $\|\hat{\theta}_0^{\text{Alg 2}} - \theta^*\|$) and the mean distance between $\hat{\theta}_u^{\text{Alg 3}}$ and θ^* (relative to $\|\hat{\theta}_0^{\text{Alg 3}} - \theta^*\|$) are both decreasing in magnitude (i.e., the iterates approach θ^*).

We now consider the same setting as in Figure 7.2 with two modifications. First, the sequences $a_k^{(1)}$ and $a_k^{(2)}$ are not required to be equal but both are assumed to have the form:

$$a_k^{(j)} = \frac{a^{(j)}}{(1 + k + A^{(j)})^{0.602}}$$

for $a^{(j)} > 0$ and $A^{(j)} \geq 0$. Second, θ is divided into only two subvectors corresponding to the first- and second halves of θ (i.e., $d = 2$). We compare the

following algorithms:

Case 1: The SPSA algorithm with $a_k = a/(1 + k + A)^{0.602}$.

Case 2: The SPSA algorithm with a_k replaced by a diagonal matrix, $\boldsymbol{A}_k$, such that the ith diagonal entry of $\boldsymbol{A}_k$ is equal to $a_k^{(j)}$ where $i \in \mathcal{S}_j$.

Case 3: Algorithm 2 (the subvector to update is selected according to a random variable) with SPSA-based gradient estimates. Each of the two subvectors is updated with equal probability.

Case 4: Algorithm 3 (the subvector to update is selected following a strictly alternating pattern) with SPSA-based gradient estimates.

For each case, the algorithms were run for a total of $T = 5000$ subvector updates and each realization was initialized at a vector following a uniform distribution on $[-5, 5]^{10}$. Thus, there were 2,500 iterations in Cases 1 and 2 (since each iteration requires updating both subvectors) and 5,000 iterations in Cases 3 and 4 (since each iteration requires updating only one subvector). We also set $c_k = c_k^{(j)} = 0.1/(1 + k)^{0.101}$ and use the same distribution for the perturbation vectors as before. Table 7.1 presents the results of part of the tuning process for Cases 1–4. Figure 7.3 shows the evolution of the loss function (for three replications) as the number of updates increases for Cases 1–4 using the best choice of parameters from Table 7.1 (for all cases the same gain sequence yielded the

smallest mean terminal loss function value). Table 7.1 and Figure 7.3 indicate that all algorithms had comparable performance. Moreover, the results support the theory on convergence of the GCSA algorithm.

7.1.2 Algorithms 2 & 3 with SG-Based Gradient Estimates

This section is concerned with an implementation of Algorithms 2 and 3 where the noisy gradient estimates are SG-based. Here, we consider the problem of linear regression with scalar output. We assume that the output, z_k, of some random process is given by $z_k = \boldsymbol{h}_k^\top \theta^* + \varepsilon_k$ for a sequence $\boldsymbol{h}_k \in \mathbb{R}^p$ of inputs and an i.i.d. sequence of mean-zero random variables ε_k. The objective is to recover the true value of θ, denoted by θ^*, using a set of input-output pairs $\{(\boldsymbol{h}_k, z_k)\}_{k\geq 1}$. An online approach to solving this problem is the following:

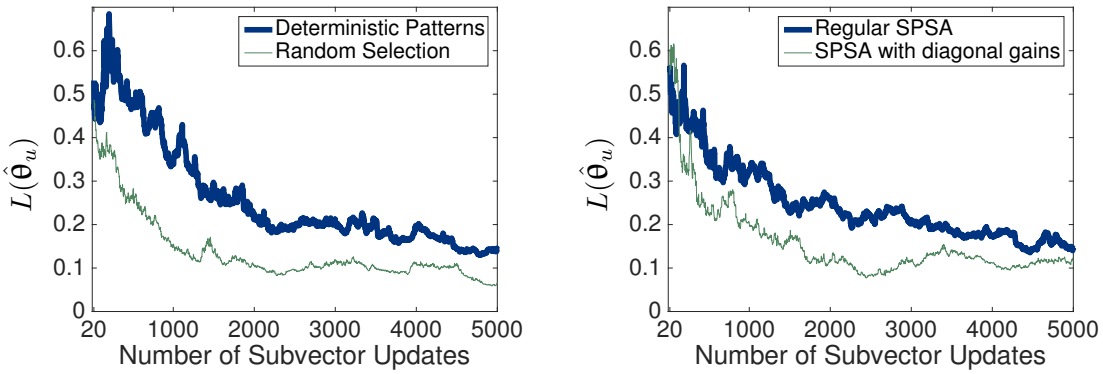
$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \left[(\boldsymbol{h}_{k+1}^\top \theta - z_{k+1})\boldsymbol{h}_{k+1}\right]_{\theta=\hat{\theta}_k}. \tag{7.2}$$

The algorithm in (7.2) is known as the least-mean-squares (LMS) algorithm.

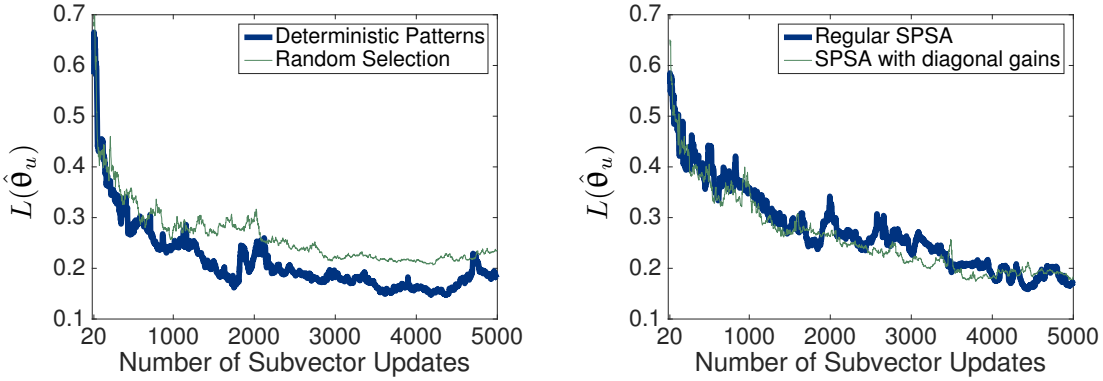
Before proceeding with the numerical experiments let us briefly motivate (7.2). First, note that if $\{\varepsilon_k\}_{k\geq 0}$ and $\{\boldsymbol{h}_k\}_{k\geq 0}$ are each i.i.d. sequences (with $\boldsymbol{h}_k$ being independent of ε_k) then, under certain conditions (e.g., Spall 2003, p. 136), the loss function $L(\theta) \equiv (1/2)E[(z_k - \boldsymbol{h}_k^\top \theta)^2]$ is independent of k, $L(\theta)$ has a unique minimum at θ^*, and the vector in square brackets in (7.2) is an unbiased measurement of the gradient of $L(\theta)$. Therefore, under certain conditions the

Algorithm	$a^{(1)}$	$a^{(2)}$	$A^{(1)}$	$A^{(2)}$	$\overline{L(\hat{\theta}_T)}$
Case 1	1	–	0	–	5.1781×10^{88}
SPSA	1	–	100	–	0.1733 *
(regular)	0.1	–	0	–	0.5291
	0.1	–	100	–	0.5613
Case 2	1	1	0	0	6.7502×10^{119}
SPSA	1	1	100	100	0.1650 *
(diagonal gain)	1	0.1	0	0	0.4335
	1	0.1	100	100	0.4236
	0.1	1	0	0	0.3941
	0.1	1	100	100	0.3691
	0.1	0.1	0	0	0.5060
	0.1	0.1	100	100	0.5727
Case 3	1	1	0	0	0.2282
SPSA	1	1	100	100	0.1982 *
(random selection)	1	0.1	0	0	0.2191
	1	0.1	100	100	0.2025
	0.1	1	0	0	0.6915
	0.1	1	100	100	0.5772
	0.1	0.1	0	0	0.6109
	0.1	0.1	100	100	0.6477
Case 4	1	1	0	0	0.1992
SPSA	1	1	100	100	0.1732 *
(deterministic patterns)	1	0.1	0	0	0.3406
	1	0.1	100	100	0.3796
	0.1	1	0	0	0.3841
	0.1	1	100	100	0.3750
	0.1	0.1	0	0	0.4812
	0.1	0.1	100	100	0.5286

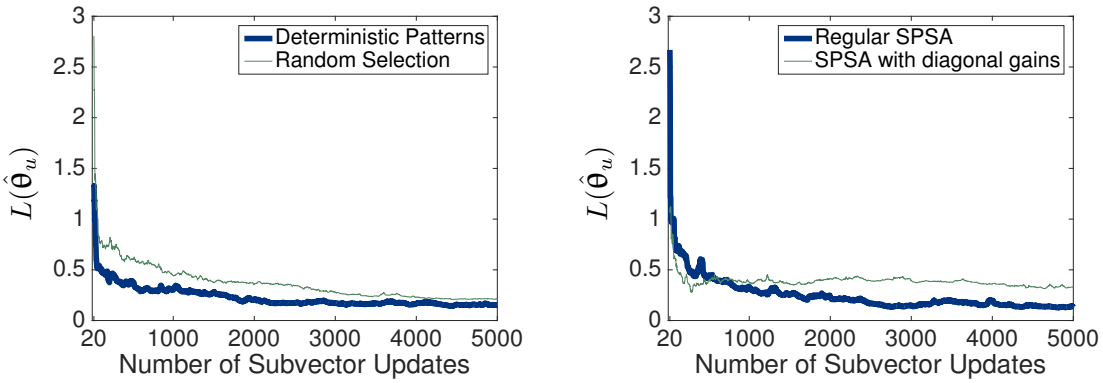
Table 7.1: Tuning the gain sequence parameters. Note that for Case 1 there is only one gain sequence, a_k, which we have set equal to $a^{(1)}/(1+k+A^{(1)})^{0.602}$. See p. 159 for a description of each of the four cases above. Here, $\overline{L(\hat{\theta}_T)}$ represents an estimate of the loss function at the terminal value obtained by averaging the terminal loss values of 100 replications. For comparison, $L(\theta^*) = 0$. An asterisk marks the smallest mean terminal loss function value for each case.



Above: the 1st replication.



Above: the 2nd replication.



Above: the 3rd replication.

Figure 7.3: Performance of Cases 1 (deterministic patterns), 2 (random selection), 3 (regular SPSA), and 4 (SPSA with diagonal gains) for three i.i.d. replications (i.e., three i.i.d. realizations of each of the four cases) using the tuned parameters from Table 7.1. The term $\hat{\theta}_u$ is a generic vector representing the vector obtained after having performed u subvector updates.

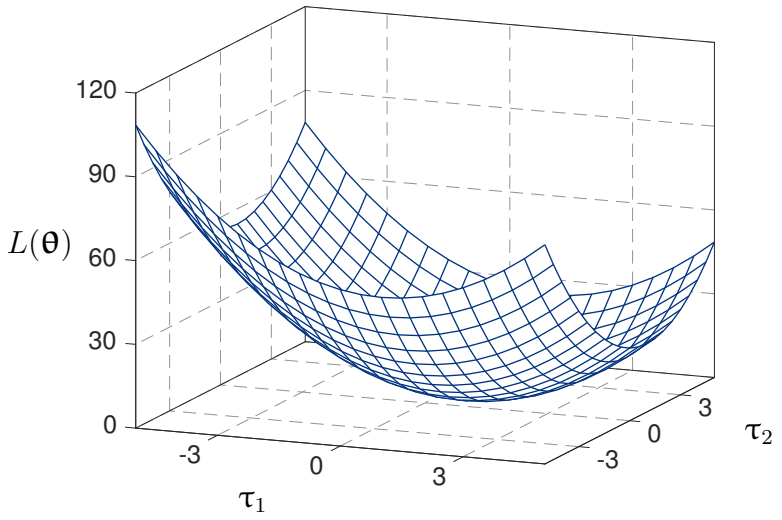


Figure 7.4: The loss function, $L(\theta) = (1/2)E[(z_k - \boldsymbol{h}_k^\top \theta)^2]$, associated with the least-mean-squares (LMS) algorithm in (7.2) for the special case where $\theta \in \mathbb{R}^2$, $\theta^* = [1,1]^\top$, the entries of $\boldsymbol{h}_k$ are uniformly distributed in $[-3,3]$, and $\varepsilon_k \sim \mathcal{N}(0,1)$ is independent of $\boldsymbol{h}_k$. Here, $L(\theta) = (1/2)[1 + 3(\theta - \theta^*)^\top(\theta - \theta^*)]$.

LMS algorithm in (7.2) is a special case of the SG algorithm (which requires an unbiased estimate of the gradient of the loss function). Figure 7.4 illustrates $L(\theta)$ for the special case where $\theta \in \mathbb{R}^2$, the entries of $\boldsymbol{h}_k$ are independent and uniformly distributed in $[-3,3]$, $\varepsilon_k \sim \mathcal{N}(0,1)$, and $\theta^* = [1,1]^\top$.

A cyclic implementation of (7.2) in the manner of Algorithms 2 and 3 would involve defining the subvectors, defining the gains sequences $a_k^{(j)}$, determining how the subvector to update in Algorithm 2 is selected, and determining how the data $\{(\boldsymbol{h}_k, z_k)\}_{k \geq 1}$ are to be generated and processed. In regards to the subvector definition, we assume $p = 10$ and use the same partition as in Section 7.1.1 (here there are 5 subvectors of length 2 each). In regards to the gain sequences, we set $a_k^{(j)} = a/(1 + k + A)^\alpha$ for all j where $a = 1$, $A = 100$, and

$\alpha = 1$. The value of α is a standard choice for the SG algorithms and the values of a and A were *lightly* tuned. Next, for Algorithm 2 each of the 5 subvectors has equal probability of being selected ($q(j) = q$ for $j = 1, \ldots, 5$). The input-output pair $(\boldsymbol{x}_k, z_k)$ was generated letting $\theta^* = [1, \ldots, 1]^\top$, $\varepsilon_k \sim \mathcal{N}(0, 1)$, and letting the entries of $\boldsymbol{h}_k$ be independent and uniformly distributed in $[-3, 3]$. Finally, each input-output pair was used to perform 5 subvector updates. Once 5 updates have been performed using the same input-output pair, a new pair is generated. Using the notation of Section 7.1.1, Figure 7.5 presents the value of $\|\hat{\theta}_u - \theta^*\|/\|\hat{\theta}_0 - \theta^*\|$ for 5 replications of Algorithms 2 and 3 initialized at $-\theta^*$. It can be seen that the iterates of both algorithms appear to converge to θ^*.

We now consider the same setting as in Figure 7.5 with three modifications. First, the gain sequences take the form:

$$a_k^{(j)} = \frac{a^{(j)}}{(1 + k + A^{(j)})^{0.501}},$$

for $a^{(j)} > 0$ and $A^{(j)} \geq 0$. Second, the vector θ is divided into only two subvectors corresponding to the first- and second halves of θ (i.e., $d = 2$). Third, we assume $\varepsilon_k \sim \mathcal{N}(0, 0.1^2)$. We compare the following algorithms:

Case 1: The SG algorithm from (7.2) with $a_k = a/(1 + k + A)^{0.501}$.

Case 2: The SG algorithm from (7.2) with a_k replaced by a diagonal matrix, $\boldsymbol{A}_k$, such that the ith diagonal entry of $\boldsymbol{A}_k$ is equal to $a_k^{(j)}$ where $i \in \mathcal{S}_j$.

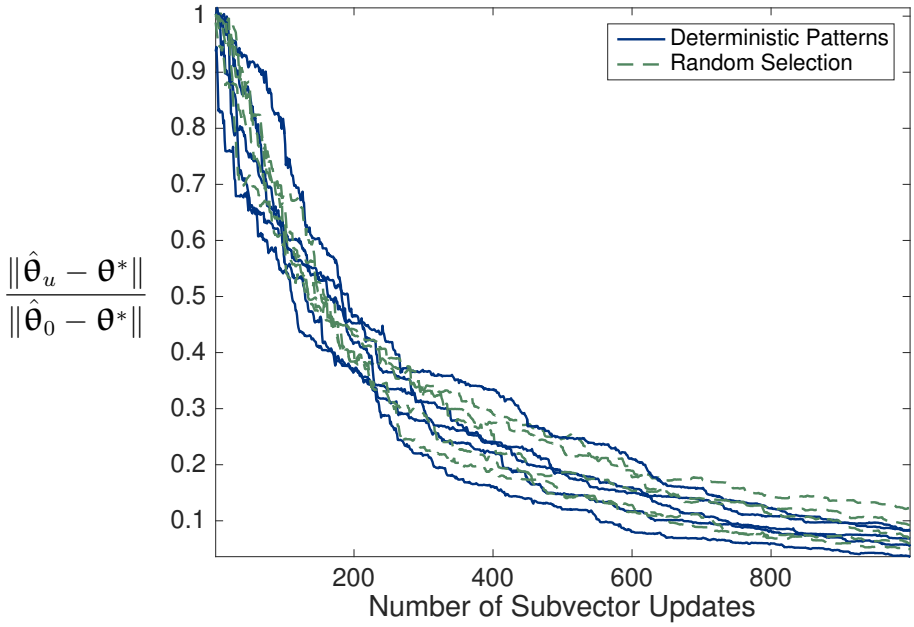


Figure 7.5: Cyclic implementations, in the manner of Algorithms 2 (random selection) and 3 (deterministic patterns), of the LMS algorithm in (7.2). Pictured are 5 realizations of each algorithm. Here, $\hat{\theta}_u$ represents either $\hat{\theta}_u^{\text{Alg 2}}$ (dashed lines) or $\hat{\theta}_u^{\text{Alg 3}}$ (solid lines).

Case 3: Algorithm 2 (the subvector to update is selected according to a random variable) with SG-based gradient estimates. Each of the two subvectors is updated with equal probability.

Case 4: Algorithm 3 (the subvector to update is selected following a strictly alternating pattern) with SG-based gradient estimates.

Each algorithm was run for a total of $T = 5000$ subvector updates and each realization was initialized at a vector uniformly distributed on $[-4, 6]^{10}$. Thus, there were 2,500 iterations in Cases 1 and 2 (since each iteration requires updating both subvectors) and 5,000 iterations in Cases 3 and 4 (since each iteration re-

quires updating only one subvector). The entries of $\boldsymbol{h}_k$ were independent and uniformly distributed in $[-3, 3]$. Table 7.2 presents the results of part of the tuning process for Cases 1–4 and Figure 7.6 shows the evolution of the loss function as the number of updates increases. Table 7.2 and Figure 7.6 indicate that all algorithms had a comparable performance (for all cases the same gain sequence yielded the smallest mean terminal loss function value). Moreover, the results support the theory on convergence of the GCSA algorithm.

7.2 Numerical Examples on Normality

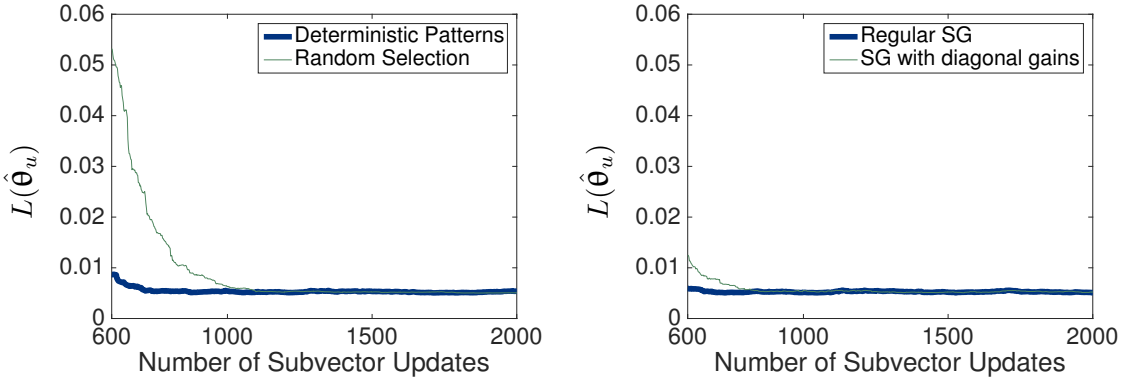
This section provides a few simple numerical examples regarding the asymptotic normality of the iterates from Algorithm 3 (where the subvector to updates is selected following a deterministic pattern). This algorithm is implemented using SG-based noisy update directions (Section 7.2.1) and SPSA-based noisy update directions (Section 7.2.2).

7.2.1 Algorithm 3 with SG-Based Gradient Estimates

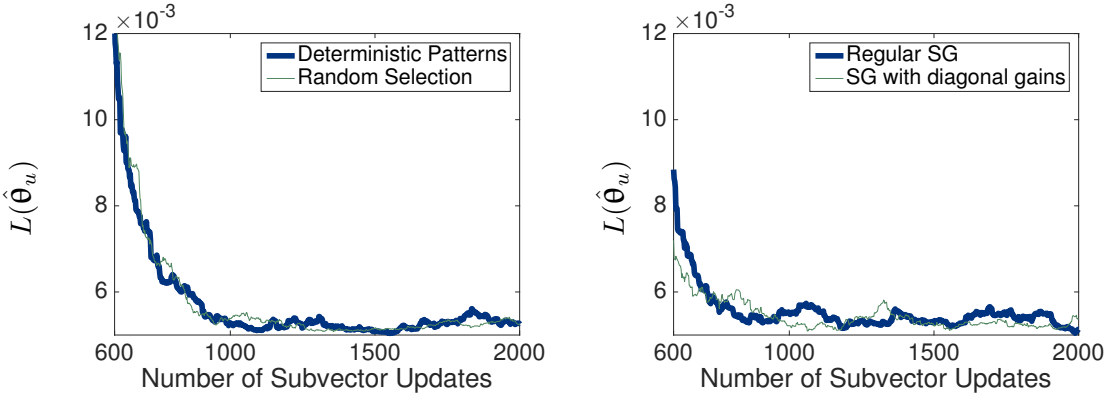
Consider the minimization of $L(\theta) = E[Q(\theta, \boldsymbol{V})|\theta]$ where $Q(\theta, \boldsymbol{V}) = \rho(\theta) + \theta^\top \boldsymbol{V}$ for some real-valued function $\rho(\theta)$ and a vector-valued random variable $\boldsymbol{V}$ that is independent of θ. Then, $L(\theta) = \rho(\theta) + \theta^\top E[\boldsymbol{V}]$. If the mean of $\boldsymbol{V}$ is unknown and $\rho(\theta)$ is continuously differentiable, an SG-based approach to solving this problem is obtained by letting $\hat{\boldsymbol{g}}_k(\hat{\theta}_k) = [\partial\rho(\theta)/\partial\theta]_{\theta=\hat{\theta}_k} + \boldsymbol{V}$. Here,

Algorithm	$a^{(1)}$	$a^{(2)}$	$A^{(1)}$	$A^{(2)}$	$\overline{L(\hat{\theta}_T)}$
Case 1	1	–	0	–	0.0073
SG	1	–	100	–	0.0072
(regular)	0.1	–	0	–	0.0052
	0.1	–	100	–	0.0052 $*$
Case 2	1	1	0	0	0.0072
SG	1	1	100	100	0.0071
(diagonal gain)	1	0.1	0	0	2.6669
	1	0.1	100	100	0.0060
	0.1	1	0	0	0.7573
	0.1	1	100	100	0.0060
	0.1	0.1	0	0	0.0052
	0.1	0.1	100	100	0.0052 $*$
Case 3	1	1	0	0	0.0064
SG	1	1	100	100	0.0063
(random selection)	1	0.1	0	0	0.0258
	1	0.1	100	100	0.0056
	0.1	1	0	0	0.0081
	0.1	1	100	100	0.0056
	0.1	0.1	0	0	0.0051
	0.1	0.1	100	100	0.0051 $*$
Case 4	1	1	0	0	1.3221×10^8
SG	1	1	100	100	0.0067
(deterministic patterns)	1	0.1	0	0	1.5906×10^{17}
	1	0.1	100	100	0.0060
	0.1	1	0	0	0.0060
	0.1	1	100	100	0.0060
	0.1	0.1	0	0	0.0051
	0.1	0.1	100	100	0.0051 $*$

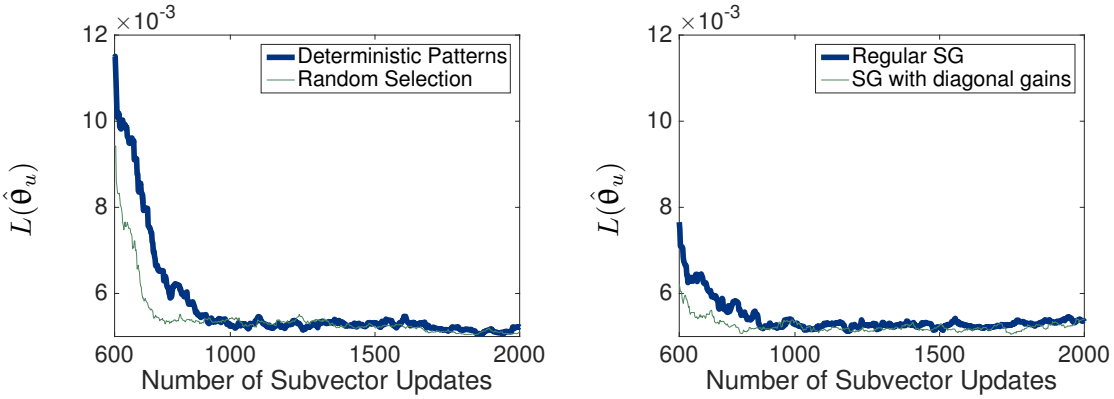
Table 7.2: Tuning the gain sequence parameters. Note that for Case 1 there is only one gain sequence, a_k, which we have set equal to $a^{(1)}/(1+k+A^{(1)})^{0.501}$. See p. 164 for a description of each of the four cases above. Here, $\overline{L(\hat{\theta}_T)}$ represents an estimate of the loss function at the terminal value obtained by averaging the terminal loss values of 100 replications. For comparison, $L(\theta^*) = 0$. An asterisk marks the smallest mean terminal loss function value for each case.



Above: the 1st replication.



Above: the 2nd replication.



Above: the 3rd replication.

Figure 7.6: Performance of Cases 1 (deterministic patterns), 2 (random selection), 3 (regular SG), and 4 (SG with diagonal gains) for three i.i.d. replications (i.e., three i.i.d. realizations of each of the four cases) using the tuned parameters from Table 7.2. The term $\hat{\theta}_u$ is a generic vector representing the vector obtained after having performed u subvector updates.

$\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$ is an unbiased estimate of the gradient of $L(\theta)$ at $\hat{\theta}_k$. If it is possible to sample from $\boldsymbol{V}$ then it is possible to obtain $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$ and update $\hat{\theta}_k$ using SG:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \left(\left[\frac{\partial \rho(\theta)}{\partial \theta} \right]_{\theta=\hat{\theta}_k} + \boldsymbol{V}_k \right), \tag{7.3}$$

where $\boldsymbol{V}_k$ denotes a measurement of $\boldsymbol{V}$. A strictly cyclic implementation of (7.3) (i.e, an implementation resembling the cyclic seesaw SG algorithm except with d subvectors) with non-overlapping subvectors is as follows:

$$\hat{\theta}_k^{(I_j)} = \hat{\theta}_k^{(I_{j-1})} - a_k^{(j)} \left(\left[\frac{\partial \rho(\theta)}{\partial \theta} \right]_{\theta=\hat{\theta}_k^{(I_{j-1})}} + \boldsymbol{V}_k^{(I_{j-1})} \right)^{(j)} \tag{7.4}$$

for $1 \leq j \leq d$, where $\hat{\theta}_k^{(I_0)} \equiv \hat{\theta}_k^{\text{cyc}}$, $\hat{\theta}_{k+1}^{\text{cyc}} \equiv \hat{\theta}_k^{(I_d)}$, and $\boldsymbol{V}_k^{(I_j)}$ is a vector-valued random variable representing a measurement of $\boldsymbol{V}$. Here, Theorem 5 could be used to derive conditions for the asymptotic normality of $k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$ where $\beta > 0$ is some constant and θ^* is a minimizer of $L(\theta)$. We do this next.

Let us discuss conditions under which C0–C8, the conditions of Theorem 5, are satisfied. First note that if $\rho(\theta)$ is twice continuously differentiable with a bounded Jacobian matrix $\boldsymbol{J}(\theta)$ and if $\hat{\theta}_k^{(I_j)} \to \theta^*$ w.p.1 for all j then condition C0 is satisfied. Next, if $a_k^{(j)} = a_k = a/(1+k+A)^\alpha$ for some $a, A > 0, \alpha > 0$ then C1 is also satisfied with $r_j = a$. Moreover, since we are assuming the d subvectors do not overlap when the matrix Γ in (5.25) is equal to $a\boldsymbol{J}(\theta^*)$. Therefore, if $\boldsymbol{J}(\theta^*)$ is positive definite then condition C2 would also be satisfied with $a\boldsymbol{J}(\theta^*) = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^\top$

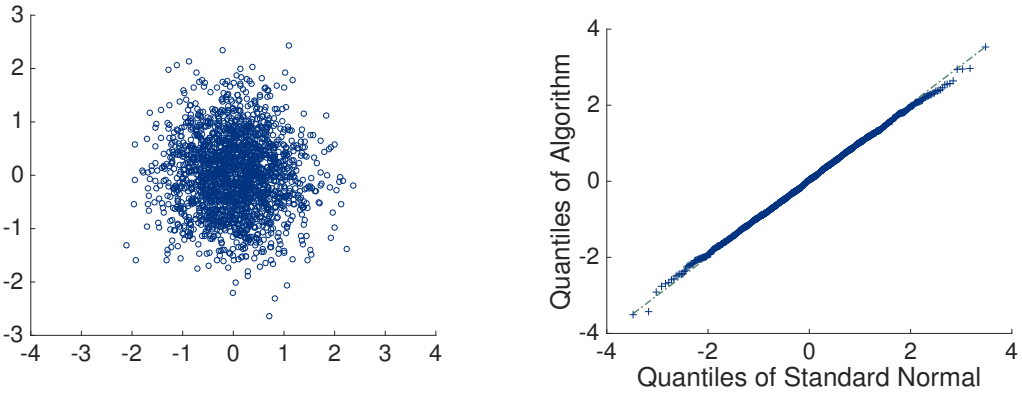
where $\boldsymbol{P} \in \mathbb{R}$ is an orthogonal matrix and $\boldsymbol{\Lambda}$ is a positive definite diagonal matrix. Next, letting $\mathcal{F}_k$ be the σ-field generated by $\{\hat{\theta}_i^{\text{cyc}}\}_{i=1}^k$, the bias term in condition C3 is identically zero (i.e., equal to the zero vector). Therefore, C3 is automatically satisfied. Finally, if the variables $\boldsymbol{V}_k^{(I_j)}$ for $j = 1, \ldots, d$ and $k \geq 0$ are i.i.d. and have (finite) positive definite covariance matrix, then conditions C4, C5-(i), C5-(iii), C6, and C7-(i) hold with $\boldsymbol{\Sigma} = a^2 \sum_{j=1}^d \operatorname{Var}\left([\boldsymbol{V}_k^{(I_j)}]^{(j)}\right)$, a positive definite block-diagonal matrix. Finally, if $0 < \alpha \leq 1$ (note that $\beta = \alpha$ by C5-(i)), then a *sufficient* condition for B6 to be satisfied is if the smallest diagonal entry of $\boldsymbol{\Lambda}$ is strictly greater than $1/2$ (if $\alpha < 1$ then this condition may be relaxed to having the smallest diagonal entry of $\boldsymbol{\Lambda}$ be strictly positive).

The discussion above gave conditions for the asymptotic normality of the vector $k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$ with $\beta = \alpha$. One important condition was the convergence w.p.1 of $\hat{\theta}_k^{(I_j)}$ to θ^*. Corollary 3 could be used to show convergence of $\hat{\theta}_k^{\text{cyc}}$ to θ^*. Then, Corollary 1 would imply the desired result. Conditions A0$''$, A3$''$, A5$''$, and A6$''$, are automatically satisfied given the assumptions of the previous paragraph. It remains to discuss the validity of A4$''$, A7$''$, and A8$''$. Condition A4$''$ requires the algorithm's iterates to be bounded w.p.1. Verifying this in practice is impossible (see Section 4.5). Next, A7$''$ requires θ^* to be a locally asymptotically stable (in the sense of Lyapunov) solution of:

$$\dot{\boldsymbol{Z}}(t) = -a\left(\left[\frac{\partial \rho(\theta)}{\partial \theta}\right]_{\theta = \boldsymbol{Z}(t)} + E[\boldsymbol{V}]\right). \tag{7.5}$$

One example of a situation in which θ^* is a locally asymptotically stable solution to (7.5) is when $\rho(\theta) = [\theta^\top \boldsymbol{H} \theta]/2$ for some positive definite matrix $\boldsymbol{H}$. Throughout the remainder of this section we will assume $\rho(\theta)$ has this form. In this case, $\dot{\boldsymbol{Z}}(t) = -a\boldsymbol{H}(\theta + \boldsymbol{H}^{-1}E[\boldsymbol{V}])$. Because $\theta^* = -\boldsymbol{H}^{-1}E[\boldsymbol{V}]$, θ^* is an equilibrium point of $\dot{\boldsymbol{Z}}(t)$. The desired Lyapunov stability follows from Lyapunov's second method for stability (e.g., Cronin 2007), and therefore A7$''$ holds. Condition A8$''$ is also impossible to verify in practice. It requires the iterates to be contained within the domain of attraction of (7.5) infinitely often w.p.1. Next we conduct a small numerical experiment to illustrate convergence and asymptotic normality under the assumptions made thus far. Aside from A4$''$ and A8$''$, all the conditions for convergence w.p.1 and asymptotic normality are satisfied.

For our numerical experiments we implement (7.4) assuming $p = 2$ (i.e., $\theta \in \mathbb{R}^2$) and that the vectors $\boldsymbol{V}_k^{(I_j)}$ are i.i.d. with $\boldsymbol{V}_k^{(I_j)} \sim \mathcal{N}(\mu, \boldsymbol{I})$, where $\mu = [5, 5]^\top$. In this setting there are two subvectors to update corresponding to the first- and second entries of θ. The matrix $\boldsymbol{H}$ was set to be equal to the identity matrix. For the gain sequences the values used were $\alpha = 0.501$, $a = 1$, and $A = 100$. Under these assumptions, $\theta^* = -[5, 5]^\top$ and $k^{0.501/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$ should converge in distribution to a random variable with distribution $\mathcal{N}(\boldsymbol{0}, 0.5\boldsymbol{I})$. Figure 7.7 gives the results for $\hat{\theta}_0^{\text{cyc}} = [1, 1]^\top$. The results in Figure 7.7a appear to support the fact that the values of $T^{0.501/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)$ are expected to have a multivariate normal distribution with a diagonal covariance matrix (this



(a) Scatter plot of $T^{\beta/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)$. **(b)** Q-Q plot of $[T^{\beta/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)]^\top \mathbf{1}$.

Figure 7.7: In the plots above the term "1" denotes the vector of ones, $\beta = 0.501$, and $T = 1000$. The plots are the result of 2,000 i.i.d. replications.

can be seen since the cluster of points in Figure 7.7a is roughly symmetric) for $T = 1000$. Additionally, because the sum of the entries of a random variable with distribution $\mathcal{N}(\mathbf{0}, 0.5\boldsymbol{I})$ should have a normal distribution with mean zero and variance 1, a Q-Q plot could be used to compare the quantiles of a standard normal random variable to the quantiles of the sum of the entries of $T^{0.501/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)$. Figure 7.7b shows the resulting Q-Q plot. It appears that the sum of the entries of the normalized iterates closely approximates a standard normal random variable as desired.

7.2.2 Algorithm 3 with SPSA-Based Gradient Estimates

This section considers the same problem from Section 7.2.1, that is the minimization of $L(\theta) = \theta^\top \boldsymbol{H}\theta/2 + \theta^\top E[\boldsymbol{V}]$. However, rather than using noisy gradient measurements of the form used in (7.4), this section relies only on noisy

measurements, $Q(\theta, \boldsymbol{V}) = \theta^\top \boldsymbol{H}\theta/2 + \theta^\top \boldsymbol{V}$, in the optimization process. This is done using the cyclic seesaw SPSA algorithm from Section 3.3. Specifically, an iteration of the algorithm is given by:

$$\hat{\theta}_k^{(I)} = \hat{\theta}_k^{\text{cyc}} - a_k^{(1)} \hat{\boldsymbol{g}}_k^{\text{SP}}(\hat{\theta}_k^{\text{cyc}}), \tag{7.6a}$$

$$\hat{\theta}_{k+1}^{\text{cyc}} = \hat{\theta}_k^{(I)} - a_k^{(2)} \hat{\boldsymbol{g}}_k^{\text{SP}}(\hat{\theta}_k^{(I)}), \tag{7.6b}$$

where $\hat{\boldsymbol{g}}_k^{\text{SP}}(\hat{\theta}_k^{\text{cyc}})$ and $\hat{\boldsymbol{g}}_k^{\text{SP}}(\hat{\theta}_k^{(I)})$ are as in (3.10) and (3.11), respectively. In order to implement the cyclic seesaw SPSA algorithm in (7.6a,b) it is necessary to specify the values of $a_k^{(j)}$, $c_k^{(j)}$, $\boldsymbol{\Delta}_k$, and $\boldsymbol{\Delta}_k^{(I)}$ (see p. 38 for the definition of the last three terms). In this section's numerical experiment the non-zero entries of $\boldsymbol{\Delta}_k$ and $\boldsymbol{\Delta}_k^{(I)}$ were set to be i.i.d. random variables taking the values ± 1 with equal probability. We also set $a_k^{(j)} = 1/(1+k+A)^\alpha$ and $c_k^{(j)} = 1/(1+k)^\gamma$ for all j, where $\alpha = 0.602$, $\gamma = 0.101$, and $A = 500$. All measurements of the random variable $\boldsymbol{V}$ are assumed to be i.i.d. with distribution $\mathcal{N}(\boldsymbol{\mu}, 0.1\boldsymbol{I})$ where $\boldsymbol{\mu}$ is the vector of fives. Next, we show convergence w.p.1 of $\hat{\theta}_k^{\text{cyc}}$ to θ^* (the minimizer of $L(\theta)$) and we derive the asymptotic distribution of $k^{\beta/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$ for $\beta = \alpha - 2\gamma$.

To prove convergence of $\hat{\theta}_k^{\text{cyc}}$ to θ^*, we verify that the conditions of Corollary 3 hold. The convergence of $\hat{\theta}_k^{(I)}$ to θ^* follows from Corollary 1. First, it is easy to see that condition A0″ holds with $r_j = 1$ for all j. Similarly, it is easy to see that condition A3″ is satisfied. Condition A4″ is impossible to guarantee in practice

and, in our numerical experiment, we do not impose any bounds forcing A4″ to hold. In order to verify A5″ and A6″ we must express the noisy gradient vectors $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})$ and $\hat{\boldsymbol{g}}_k^{\mathrm{SP}}(\hat{\boldsymbol{\theta}}_k^{(I)})$ in terms of their bias- and noise terms given in (3.12a,b) and (3.13a,b), respectively. In our example $\boldsymbol{\beta}_k^{(1)}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}}) = \boldsymbol{\beta}_k^{(1)}(\hat{\boldsymbol{\theta}}_k^{(I)}) = \mathbf{0}$ (due to the fact that the loss function is quadratic in $\boldsymbol{\theta}$) and the mth entry of the noise term $\boldsymbol{\xi}_k^{(1)}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})$ is equal to:

$$\left[\boldsymbol{\xi}_k^{(1)}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})\right]^{[m]} = \frac{\boldsymbol{\Delta}_k^{\top}(\boldsymbol{V}_k^{+} + \boldsymbol{V}_k^{-})}{2\Delta_k^{[m]}} + \frac{\boldsymbol{\Delta}_k^{\top}\boldsymbol{H}\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}}}{\Delta_k^{[m]}} + \frac{(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})^{\top}(\boldsymbol{V}_k^{+} - \boldsymbol{V}_k^{-})}{2c_k\Delta_k^{[m]}} - \left[\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})\right]^{[m]}$$

for $m \in \mathcal{S}_1$ (see p. 44 for the definition of $\mathcal{S}_j$) and $[\boldsymbol{\xi}_k^{(1)}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})]^{[m]} = 0$ otherwise. Similarly, the mth entry of the noise term $\boldsymbol{\xi}_k^{(2)}(\hat{\boldsymbol{\theta}}_k^{\mathrm{cyc}})$ is equal to:

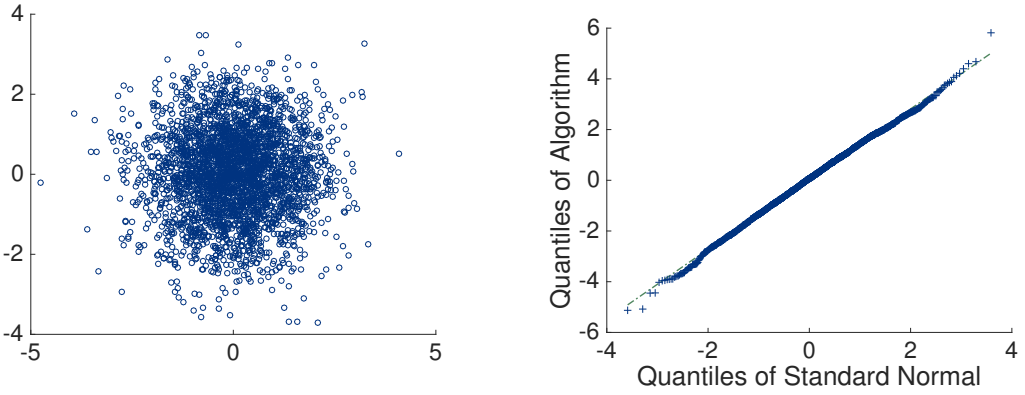
$$\left[\boldsymbol{\xi}_k^{(2)}(\hat{\boldsymbol{\theta}}_k^{(I)})\right]^{[m]} = \\ \frac{(\boldsymbol{\Delta}_k^{(I)})^{\top}(\boldsymbol{V}_k^{(I)+} + \boldsymbol{V}_k^{(I)-})}{2(\boldsymbol{\Delta}_k^{(I)})^{[m]}} + \frac{(\boldsymbol{\Delta}_k^{(I)})^{\top}\boldsymbol{H}\hat{\boldsymbol{\theta}}_k^{(I)}}{(\boldsymbol{\Delta}_k^{(I)})^{[m]}} + \frac{(\hat{\boldsymbol{\theta}}_k^{(I)})^{\top}(\boldsymbol{V}_k^{(I)+} - \boldsymbol{V}_k^{(I)-})}{2c_k(\boldsymbol{\Delta}_k^{(I)})^{[m]}} - \left[\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k^{(I)})\right]^{[m]}$$

for $m \in \mathcal{S}_2$ and $[\boldsymbol{\xi}_k^{(2)}(\hat{\boldsymbol{\theta}}_k^{(I)})]^{[m]} = 0$ otherwise. Given that the bias terms are identically zero, condition A5″ is satisfied. Using the independence of the $\boldsymbol{\Delta}$ and $\boldsymbol{V}$ and the fact that both these variables have finite variance, condition A6″ follows from the discussion on p. 92 (using the fact that $\sum_{k=1}^{\infty} a_k^2/c_k^2 < \infty$). The validity of condition A7″ has already been shown in Section 7.2.1. Finally, condition A8″, like condition A4″, is impossible to verify. Aside from conditions A4″ and A8″, all the conditions for convergence w.p.1 hold. Next, we verify the

conditions for the asymptotic normality of $k^{(\alpha-2\gamma)/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$ and compute the parameters of the asymptotic distribution.

To prove the desired result on asymptotic normality we show that the conditions of Theorem 5 are satisfied. For simplicity, we will assume $\theta \in \mathbb{R}^2$. The validity of C0–C3 has already been shown (see Section 7.2.1 and the previous paragraph). Condition C4 follows immediately by construction. Let us now show that condition C5-(iv) holds. First, because both noise terms have mean zero (conditionally on $\mathcal{F}_k$, where $\mathcal{F}_k$ was defined in Section 7.2.1), the conditional covariance matrix of $\xi_k^{(1)}(\hat{\theta}_k)$ and $\xi_k^{(2)}(\hat{\theta}_k^{(I)})$ is $E[\xi_k^{(1)}(\hat{\theta}_k)\xi_k^{(2)}(\hat{\theta}_k^{(I)})^\top|\mathcal{F}_k]$. Using the convergence (w.p.1) of the iterates to $\hat{\theta}_k^*$ along with the continuity of the gradient of $L(\theta)$ it is easy to show that $E[\xi_k^{(1)}(\hat{\theta}_k)\xi_k^{(2)}(\hat{\theta}_k^{(I)})^\top|\mathcal{F}_k] = O(1/c_k)$ w.p.1. However, $c_k = O(1/k^\gamma)$ which implies $\lim_{k\to\infty} k^{\beta-\alpha}/c_k = 0$. Therefore, $k^{\beta-\alpha}E[\xi_k^{(1)}(\hat{\theta}_k)\xi_k^{(2)}(\hat{\theta}_k^{(I)})^\top|\mathcal{F}_k] \to 0$ w.p.1. Next we verify the remaining conditins for asymptotic normality.

After a direct calculation of the conditional covariance matrices of the two noise terms it follows that C5-(ii) and C6 hold with $\Sigma = (0.1/4)(\theta^*)^\top\theta^* \boldsymbol{I}$. Condition C7–(ii) follows form the uniform integrability of the weighted (by $k^{\beta-\alpha}$) noise terms. Condition C8 follows from the fact that $\boldsymbol{H}$ is positive definite. If we assume $\boldsymbol{H} = \boldsymbol{I}$ for simplicity, the asymptotic distribution of $k^{(\alpha-2\gamma)/2}(\hat{\theta}_k^{\text{cyc}} - \theta^*)$ according to Theorem 5 is then $\mathcal{N}(\mathbf{0}, (5/8)\boldsymbol{I})$. Furthermore, $(8/10)^{1/2}[T^{\beta/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)]^\top\mathbf{1} \sim \mathcal{N}(0,1)$. Figure 7.8 gives the results for $\hat{\theta}_0^{\text{cyc}} = [1,1]^\top$. The results in



(a) Scatter plot of $T^{\beta/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)$. **(b)** Q-Q plot of $(8/10)^{1/2}[T^{\beta/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)]^\top \mathbf{1}$.

Figure 7.8: In the plots above the term "1" denotes the vector of ones, $\beta = \alpha - 2\gamma = 0.4$, and $T = 3000$. The plots are the result of 3,000 i.i.d. replications.

Figure 7.8a appears to support the fact that the values of $T^{(\alpha-2\gamma)/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)$ are expected to have a multivariate normal distribution with a diagonal covariance matrix (this can be seen since the cluster of points in Figure 7.8a is roughly symmetric) for $T = 3000$. Moreover, Figure 7.8b shows that $(8/10)^{1/2}[T^{\beta/2}(\hat{\theta}_T^{\text{cyc}} - \theta^*)]^\top \mathbf{1}$ appears to have a standard normal distribution as desired.

7.3 Numerical Analysis on Efficiency

This section contains simulation-based estimates of the relative efficiency ratio (6.15) for comparing the SG- and cyclic seesaw SG algorithms when c^{cyc} and c^{non} are defined as in Section 6.1 (this is done in Section 7.3.1) and for comparing the SPSA- and cyclic seesaw SPSA algorithms under a few different definitions of c^{cyc} and c^{non} (this is done in Section 7.3.2).

7.3.1 Efficiency: SG versus Cyclic Seesaw SG

This section compares the SG algorithm to the cyclic seesaw SG algorithm by estimating the relative efficiency given by the ratio in (6.15). The SG algorithm treated is a distributed version of the LMS algorithm in (7.2) where measurements of the pair $(\boldsymbol{h}_k, z_k)$ with $z_k = \boldsymbol{h}_k^\top \theta^* + \varepsilon_k$ are used to estimate θ^*. The non-cyclic implementation of LMS considered is the following:

$$[\hat{\theta}_{k+1}^{\text{non}}]^{(1)} = [\hat{\theta}_k^{\text{non}}]^{(1)} - a_k^{(1)} \left[(\boldsymbol{h}_{k+1}^\top \hat{\theta}_k^{\text{non}} - z_{k+1}) \boldsymbol{h}_{k+1} \right]^{(1)}, \tag{7.7a}$$

$$[\hat{\theta}_{k+1}^{\text{non}}]^{(2)} = [\hat{\theta}_k^{\text{non}}]^{(2)} - a_k^{(2)} \left[(\boldsymbol{h}_{k+1}^\top \hat{\theta}_k^{\text{non}} - z_{k+1}) \boldsymbol{h}_{k+1} \right]^{(2)}, \tag{7.7b}$$

where $\theta \in \mathbb{R}^p$ for $p = 12$ even and $p' = p/2 = 6$ is the subvector length. In the distributed implementation considered here, (7.7a) and (7.7b) are computed at physically different locations. To highlight the distributed nature of the implementation, throughout the remainder of this section we use the multi-agent analogy from Section 6.1 (see p. 131). In other words, we assume that each of the two lines in (7.7a,b) is computed by a different agent.

The cyclic seesaw counterpart to (7.7a,b), which is also assumed to be implemented in a distributed manner, is defined as follows:

$$\hat{\theta}_k^{(I)} = \hat{\theta}_k^{\text{cyc}} - a_k^{(1)} \left[(\boldsymbol{h}_{k+1}^\top \hat{\theta}_k^{\text{cyc}} - z_{k+1}) \boldsymbol{h}_{k+1} \right]^{(1)}, \tag{7.8a}$$

$$\hat{\theta}_{k+1}^{\text{cyc}} = \hat{\theta}_k^{(I)} - a_k^{(2)} \left[(\boldsymbol{h}_{k+1}^\top \hat{\theta}_k^{(I)} - z_{k+1}) \boldsymbol{h}_{k+1} \right]^{(2)}. \tag{7.8b}$$

Once again, we use the multi-agent analogy from Section 6.1 (see p. 131) to highlight the fact that the cyclic seesaw algorithms is assumed to be implemented in a distributed manner. Note that $(\boldsymbol{h}_{k+1}, z_{k+1})$ is used to perform two subvector updates in (7.8a,b): once to obtain $\hat{\theta}_k^{(I)}$ and once to obtain $\hat{\theta}_{k+1}^{\text{cyc}}$. The pair $(\boldsymbol{h}_{k+1}, z_{k+1})$ is also used to perform two subvector updates in (7.7a,b).

7.3.1.1 Cost as a Measure of Arithmetic Computations

The first definition of cost we consider for comparing (7.7a,b) to (7.8a,b) via (6.1) is the arithmetic cost of Section 6.1. Here, the per-iteration costs c^{non} and c^{cyc} denote the number of basic arithmetic operations required to obtain $\hat{\theta}_{k+1}^{\text{non}}$ from $\hat{\theta}_k^{\text{non}}$ and to obtain $\hat{\theta}_{k+1}^{\text{cyc}}$ from $\hat{\theta}_k^{\text{cyc}}$, respectively. In order to compute these quantities it is necessary to specify the computational graphs (CGs) associated with the algorithms, these are given in Figures 7.9 and 7.10. From Figure 7.9 we see that the per-iteration cost associated with the SG algorithm is $c^{\text{non}} = 4p+3$ operations ($3p+1$ operations from the first graph and $p+2$ from the second). In contrast, Figure 7.10 implies $c^{\text{cyc}} = 5p+3$ operations ($3p+1$ operations from the first graph and $2p+2$ from the second).

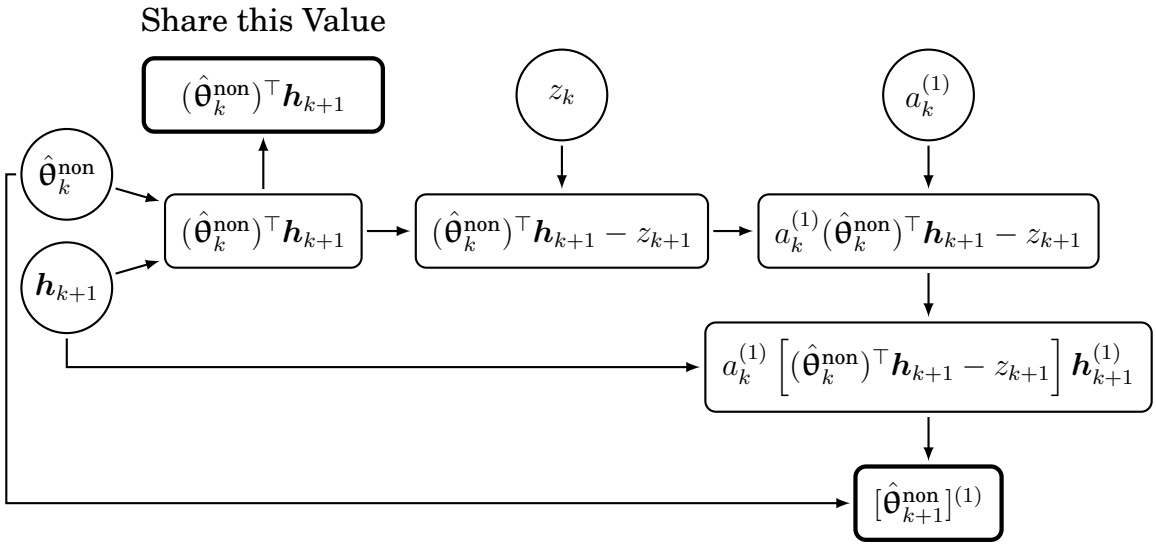
In our numerical examples we use $p = 12$ so that $c^{\text{non}} = 51$, $c^{\text{cyc}} = 63$, and $c^{\text{cyc}}/c^{\text{non}} = 21/17 \approx 5/4$. The significance of this ratio, as was explained in the discussion following Proposition 3, is that the cost of computing $\hat{\theta}_{4i}^{\text{cyc}}$ is approximately equal to the cost of computing $\hat{\theta}_{5i}^{\text{non}}$ for any strictly-positive integer i.

Therefore, for every four iterations of the cyclic seesaw SG algorithm in (7.8a,b) we may run five iterations of the SG algorithm in (7.7a,b) at the same computational cost. Consequently, $k_1 = 4i$ and $k_2 = 5i$ are valid values for k_1 and k_2 in (6.1) for any strictly-positive integer i. Figure 7.11 estimates the relative efficiency ratio in (6.1) with $k_1 = 4i$ and $k_2 = 5i$. The estimates were obtained by using $\theta^* = [1, \ldots, 1]^\top$, letting the entries of $\boldsymbol{h}_k$ be independent and uniformly distributed in $[-3, 3]$, $\epsilon_k \sim \mathcal{N}(0, 0.5^2)$, and setting $a_k = a_k^{(j)} = 0.1/(1 + k + 100)$. The expectations in (6.1) were estimated using the average of 100 i.i.d. replications. For each replication, both algorithms were initialized at $-\theta^*$.

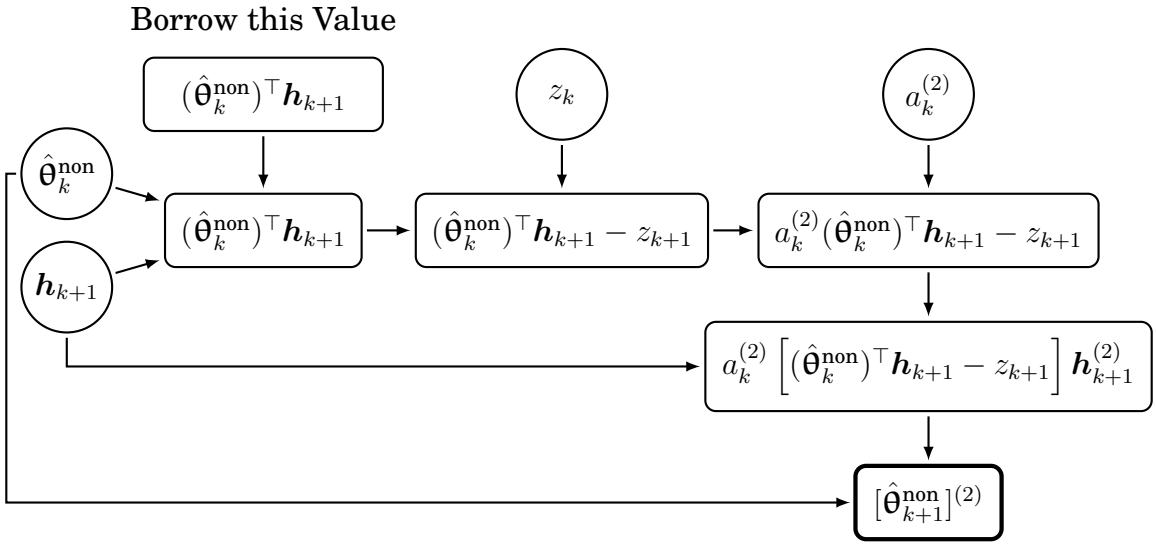
Figure 7.11 shows that the cyclic implementation was asymptotically less efficient (by approximately 15%). While the cyclic algorithm was more efficient for small values of i (small values of i correspond to earlier iterations of the algorithms), the relative efficiency for small i was found to be highly sensitive to the initialization of the algorithms while the asymptotic relative efficiency appeared to be more robust to initialization; this pattern was observed based on 10 other initializations of the algorithms.

7.3.1.2 Cost as the Number of Subvector Updates

The second type of cost we consider is the number of subvector updates performed. Under this definition of cost the SG algorithm and its cyclic counterpart have exactly the same cost since two subvector updates are performed

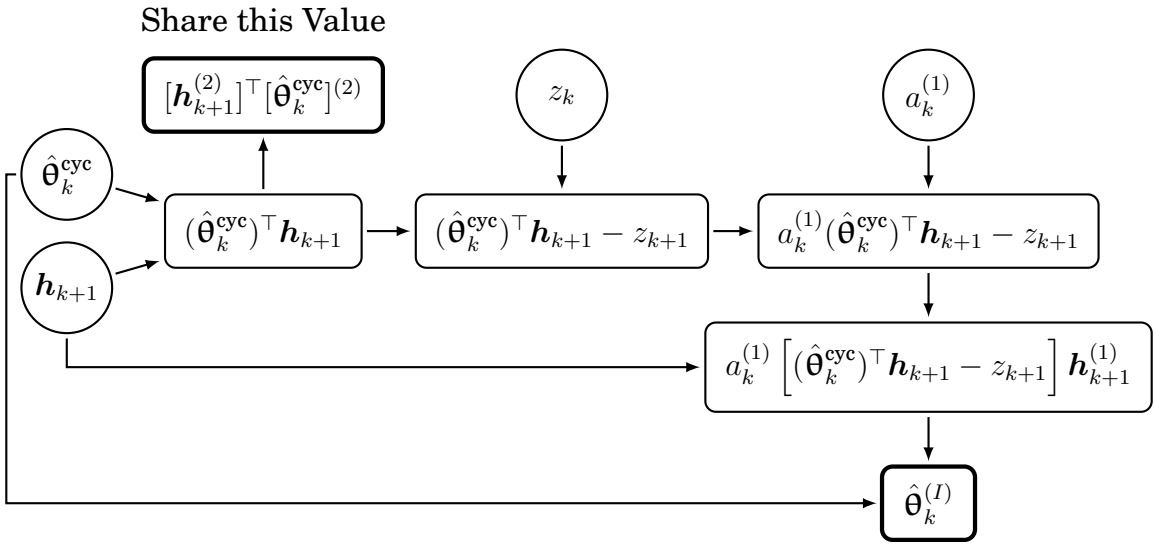


The CG for computing (7.7a).

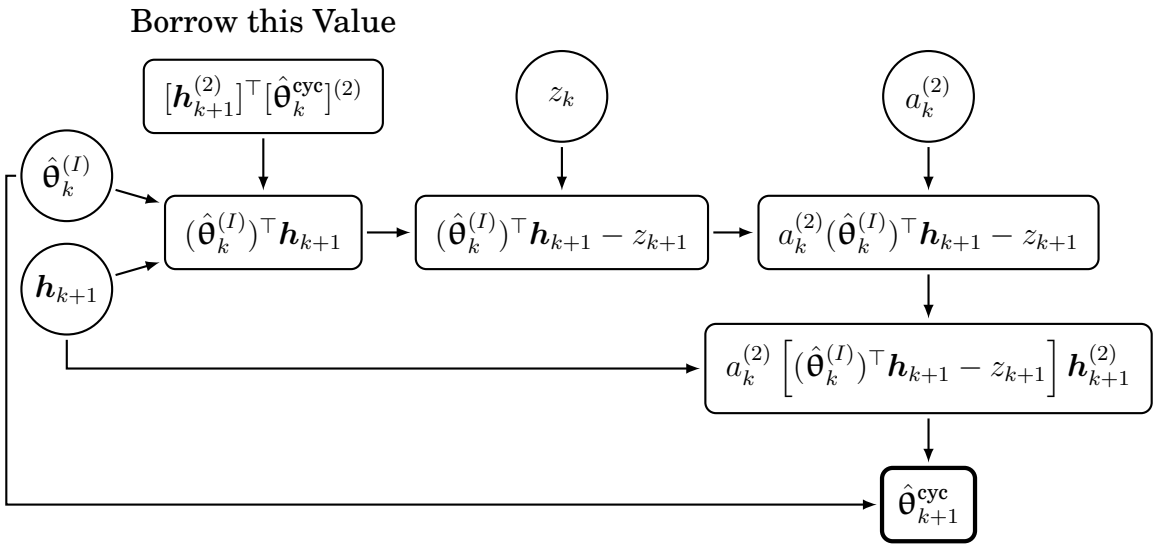


The CG for computing (7.7b).

Figure 7.9: CGs for computing the distributed SG update in (7.7a,b). In the distributed setting considered (i.e., in the multi-agent setting) the first graph, respectively the second graph, corresponds to computations performed by Agent 1, respectively Agent 2. Note that we are assuming both agents have access to $\hat{\theta}_k^{\text{non}}$, $\boldsymbol{h}_{k+1}$, and z_k. In general these CGs are not unique.



The CG for computing (7.8a).



The CG for computing (7.8b).

Figure 7.10: CGs for computing the distributed cyclic seesaw SG update in (7.8a,b). In the multi-agent setting the first graph, respectively the second graph, corresponds to computations performed by Agent 1, respectively Agent 2. Note that we are assuming both agents have access to $\boldsymbol{h}_{k+1}$ and z_k. We are also assuming Agent 1 has access to $\hat{\theta}_k^{\text{cyc}}$ and that Agent 2 has access to $\hat{\theta}_k^{(I)}$. In general these CGs are not unique.

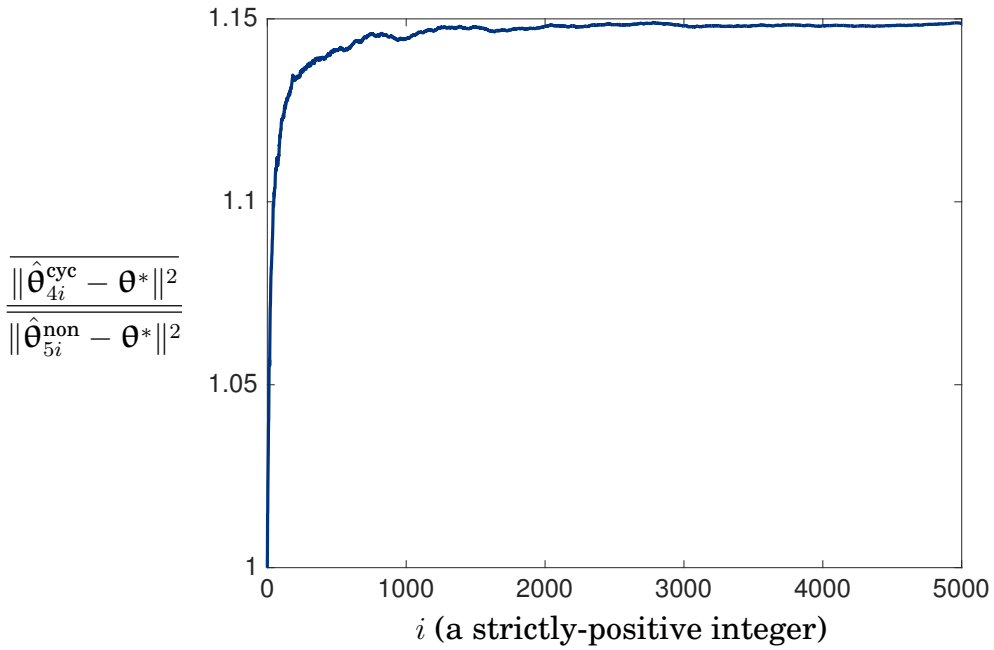


Figure 7.11: An estimate of the relative efficiency in (6.1) for comparing the cyclic seesaw SG algorithm to the regular (i.e., non-cyclic) SG algorithm when cost is a measure of the number of arithmetic computations.

every iteration. In other words, when cost is defined as the per-iteration number of subvector updates, it follows that $c^{\text{non}} = c^{\text{cyc}}$. Therefore, $k_1 = i = k_2$ are valid values for k_1 and k_2 in (6.1) for any strictly-positive integer i. Figure 7.12 estimates (6.1) for multiple values of i using the same settings from Section 7.3.1.1. The expectations in (6.1) were estimated using the average of 100 i.i.d. replications. For each replication, both the cyclic and non-cyclic algorithms were initialized at $-\theta^*$. Asymptotically, the cyclic algorithm was 5% less efficient. Based on 10 other initializations of the algorithms, it was observed that the relative efficiency was more sensitive to initialization for early iterations than for later iterations.

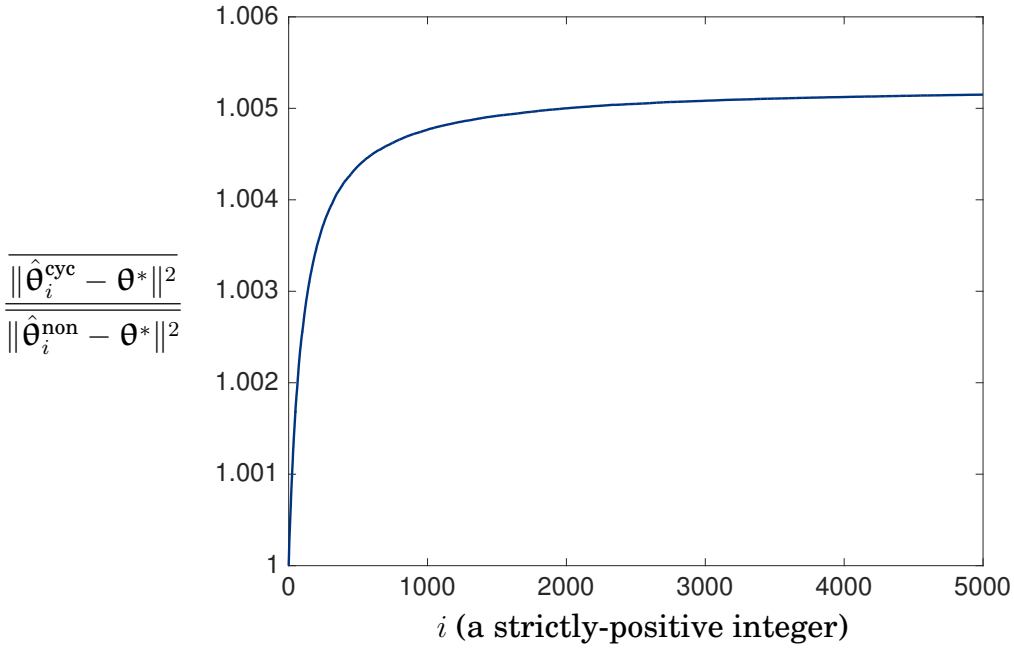


Figure 7.12: An estimate of the relative efficiency in (6.1) for comparing the cyclic seesaw SG algorithm to the regular (i.e., non-cyclic) SG algorithm when cost is a measure of the number of subvector updates.

7.3.1.3 Cost as Time allowing for Agent Unavailability

The efficiency analysis in this section once again relies on drawing parallels between the distributed implementations of (7.7a,b) and (7.8a,b) and the multi-agent setting. In the multi-agent setting, the first and second subvectors of θ correspond, respectively, to parameters that Agent No. 1 and Agent No. 2 can update. Implementing the SG algorithm in (7.7a,b) in a distributed manner would require both agents to be available at the same time. However, in practice it is not uncommon for agents to be tasked with secondary objectives that may make them temporarily unavailable to update their parameter vector. A

slight modification to the cyclic seesaw SG algorithm in (7.8a,b) would be to allow a single agent, while available, to continuously update its parameters until the moment when the other agent becomes available. The resulting algorithm would be a special case of Algorithm 1. This section constructs a simple model of agent availability in order to estimate (6.1) when cost is a measure of time.

To model agent availability we begin by defining a unit of time as the time it takes to perform a subvector update, this time is assumed to be equal for all subvectors independently of whether we are implementing the SG algorithm or its modified cyclic version and independently of the magnitude of the update (note that for the regular SG algorithm the time needed to perform a single iteration is one time unit since the two subvector updates are performed simultaneously). We will assume that any agent can become unavailable with probability q after completing its latest subvector update (for simplicity we assume an agent cannot become unavailable while in the middle of performing an update). Furthermore, at the next time unit an agent that was previously unavailable will remain unavailable with probability q. Because implementing the SG algorithm in (7.7a,b) requires both agents to be available simultaneously, no updates can be performed using the non-cyclic algorithm if at least one of the agents is unavailable. In contrast, the proposed modification to the cyclic algorithm (where an available agent continuously updates its parameters) could still be implemented provided at least one of the agents is available.

Given our definition of a time unit and our model of agent unavailability, the relative efficiency ratio in (6.1) is estimated as the ratio:

$$\frac{E\|\hat{\theta}^{\text{cyc}}(t) - \theta^*\|^2}{E\|\hat{\theta}^{\text{non}}(t) - \theta^*\|^2}, \tag{7.9}$$

where t denotes time and where $\hat{\theta}^{\text{cyc}}(t)$ and $\hat{\theta}^{\text{non}}(t)$ denote algorithm iterate values at time t (we deliberately avoid writing t as a subscript of $\hat{\theta}$ due to the fact that subscripts in (7.7a,b) and (7.8a,b) are reserved for the iteration number, which is not the same as t). In order to select a value of q for our numerical examples we note the following. At any given time the probability of updating the parameter vector using the SG algorithm is $P_1 \equiv (1-q)^2$ whereas the probability of updating the parameter vector using the modified cyclic seesaw algorithm is $P_2 \equiv 2(1-q) - (1-q)^2$. For $q \in [0,1]$, $P_2 \geq P_1$ and the maximum difference between P_1 and P_2 occurs at $q = 1/2$. Figure 7.13 estimates (7.9) using $q = 1/2$ with $a_k = a_k^{(j)} = 0.1/(1+100+k)^{0.501}$. The expectations in (7.9) were estimated using the average of 100 i.i.d. replications. For each replication, both algorithms were initialized at $-\theta^*$. We use the same distributions for $\boldsymbol{h}_k$ and ε_k from Section 7.3.1.1. It was assumed that a new input-output pair $(\boldsymbol{h}_k, z_k)$ became available with the passing of every time unit. Agents were assumed to use the latest input-output pair to perform their updates. The results of Figure 7.13 indicate that the cyclic variant was approximately 10% more effi-

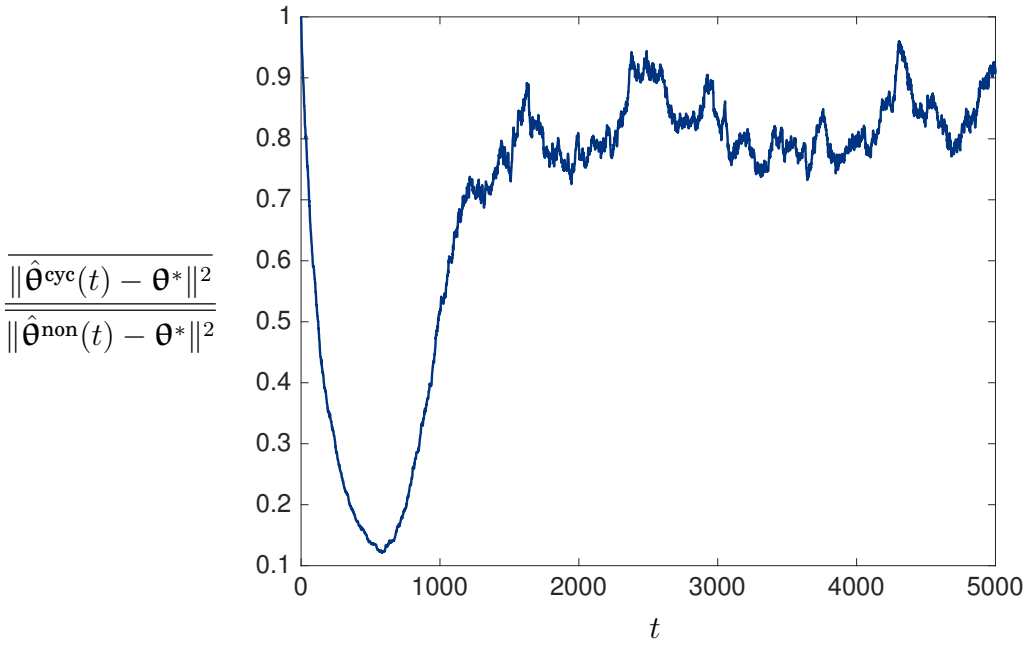


Figure 7.13: An estimate of the relative efficiency in (7.9) for comparing the SG to a generalized cyclic seesaw SG algorithm where agents may be unavailable to perform updates with probability $q = 1/2$.

cient (asymptotically), likely due to the fact that using a cyclic implementation implies the parameter vector is updated more frequently than when using a non-cyclic implementation. Based on 10 other initializations of the algorithms, it was observed that the relative efficiency was more sensitive to initialization for small values of t than for larger values.

7.3.2 Efficiency: SPSA versus Cyclic Seesaw SPSA

This section compares the SPSA algorithm to the cyclic seesaw SPSA algorithm by estimating the relative efficiency in (6.1). The SPSA algorithm treated is the SPSA algorithm from Section 7.1.1 where the objective is to minimize the

skewed quartic loss function with $p = 10$. We assume $Q(\boldsymbol{\theta}, \boldsymbol{V}) = L(\boldsymbol{\theta}) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 0.1^2)$. For the cyclic seesaw implementation we let $p' = 5$ and use noisy gradient estimates of the form in (3.10) and (3.11), SPSA-based noisy gradient estimates where only the subvector to update is perturbed.

7.3.2.1 Cost as the Number of Noisy Loss Measurements

The first definition of cost we consider is the number of noisy loss function measurements required per-iteration. Under this definition $c^{\text{non}} = 2$ for the the regular SPSA algorithm whereas $c^{\text{cyc}} = 4$ for the cyclic seesaw SPSA algorithm. Therefore, in order to conduct a fair comparison between SPSA and cyclic SPSA we would need to run twice as many iterations for the regular SPSA algorithm. Therefore, $k_1 = i$ and $k_2 = 2i$ are valid values for k_1 and k_2 in (6.1) for any strictly-positive integer i. As such, Figure 7.14 estimates the relative efficiency in (6.1) with $k_1 = i$ and $k_2 = 2i$ for different values of i. The expectations in (6.1) were estimated using the average of 500 i.i.d. replications. For each replication, both algorithms were initialized at $-[1, \ldots, 1]^\top$. As suggested by Table 7.1, we use the gain sequences $a_k = a_k^{(j)} = 1/(1 + k + 100)^{0.602}$ and set $c_k = c_k^{(j)} = 0.1/(1 + k)^{0.101}$. We let the non-zero entries of the $\boldsymbol{\Delta}$ be independent and take the value ± 1 with equal probability. We can see that the relative efficiency favors the cyclic algorithm for early iterations (although this behavior was, once again, sensitive to the initialization of the algorithms) while the non-cyclic

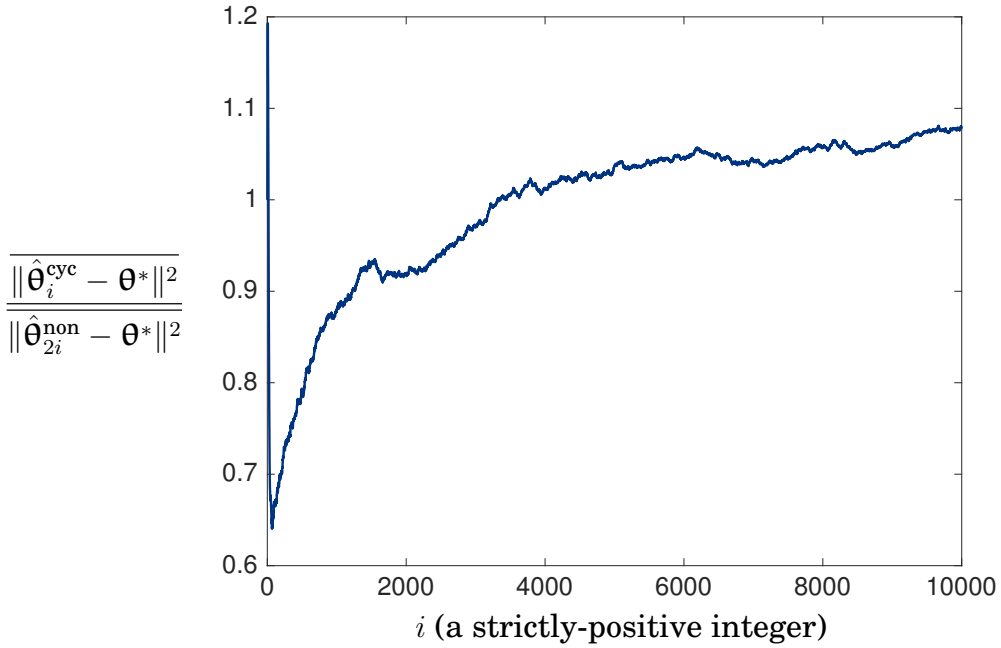


Figure 7.14: An estimate of the relative efficiency in (6.1) for comparing the cyclic seesaw SPSA algorithm to the regular (i.e., non-cyclic) SPSA algorithm when cost is a measure of the number of noisy loss function evaluations.

algorithm becomes more efficient (by about 8%) in the long run. Based on 10 different initializations of the other, it was observed that the relative efficiency was more sensitive to initialization for small values of t than for larger values.

7.3.2.2 Cost as the Number of Subvector Updates

This section estimates the ratio in (6.1) by viewing cost as the number of subvector updates per-iteration, which implies $c^{\text{non}} = c^{\text{cyc}} = 2$. Therefore, $k_1 = i = k_2$ are valid values for k_1 and k_2 in (6.1) for any strictly-positive integer i. Figure 7.15 presents the results with $k_1 = i = k_2$ using the settings from Section 7.3.2.1. The expectations in (6.1) were estimated using the aver-

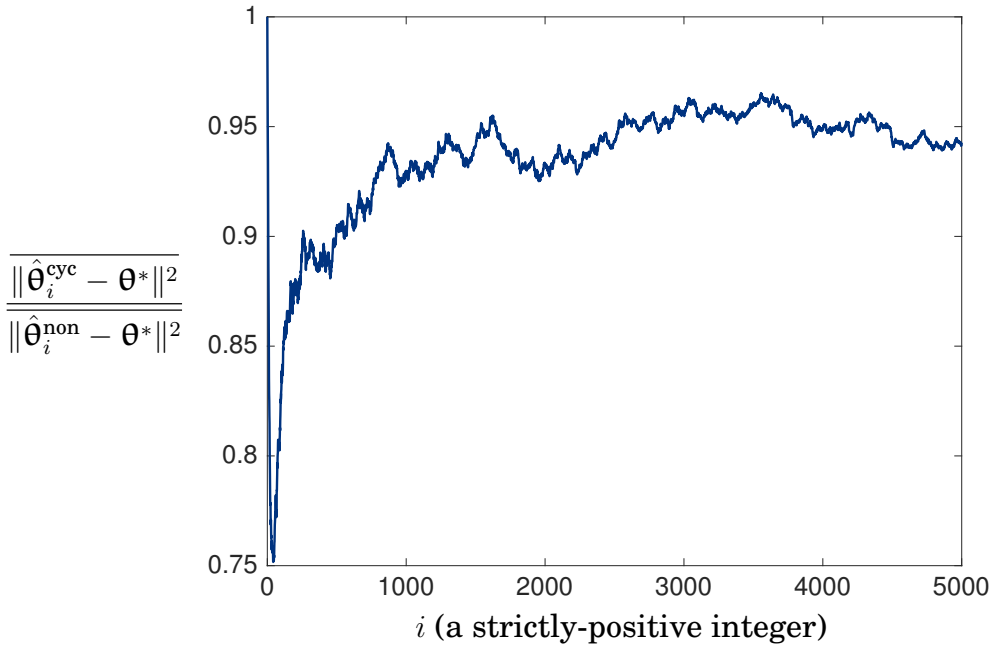


Figure 7.15: An estimate of the relative efficiency in (6.1) for comparing the cyclic seesaw SPSA algorithm to the regular (i.e., non-cyclic) SPSA algorithm when cost is a measure of the number of subvector updates.

age of 500 i.i.d. replications. For each replication, both the cyclic and non-cyclic algorithms were initialized at $-[1, \ldots, 1]^\top$. It can be seen that cyclic algorithm slightly outperforms its non-cyclic counterpart (by about 6%) when the number of iterations is large. Based on 10 other initializations of the algorithms, the relative efficiency for small values of i appeared to be more sensitive to initialization than for large values of i.

7.3.2.3 Cost as Time allowing for Agent Unavailability

Similarly to Section 7.3.1.3, this section estimates relative efficiency in a multi-agent setting where agents can be unavailable with probability $q = 1/2$.

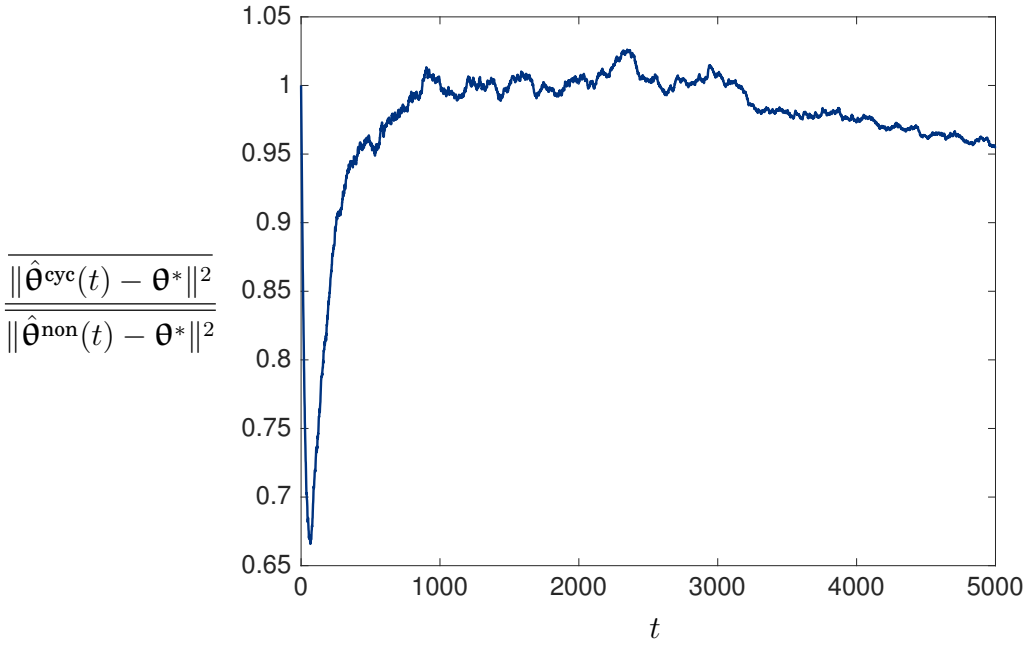


Figure 7.16: An estimate of the relative efficiency in (7.9) for comparing the SPSA to a generalized cyclic seesaw SPSA algorithm where agents may be unavailable to perform updates with probability $q = 1/2$.

The difference between the experiment in this section and the experiment from Section 7.3.1.3 is that SPSA-based gradient estimates, instead of SG-based estimates, are used to perform the subvector updates. For this reason we refer the reader to Section 7.3.1.3 for a more detailed description of the model for agent unavailability. Figure 7.16 presents the estimate of (7.9) as a function of t, denoting time, using the settings from Section 7.3.2.1. The expectations in (7.9) were estimated using the average of 500 i.i.d. replications. For each replication, both algorithms were initialized at the vector $-[1, \ldots, 1]^\top$. We see that asymptotically the cyclic algorithm appeared to be only slightly more efficient than its non-cyclic counterpart (based on 10 other initializations, this appeared

to be independent of the initialization of the cyclic and non-cyclic algorithms).

7.4 Concluding Remarks

This chapter contains numerical results illustrating the theory on convergence from Chapter 4 and the theory on asymptotic normality from Chapter 5. Additionally, numerical experiments were conducted to investigate the relative efficiency (as defined in (6.1)) between cyclic algorithms and their non-cyclic counterparts. For the numerical experiments that were conducted, the cyclic and non-cyclic algorithms had comparable asymptotic efficiency. However, it is important to keep in mind that there are many factors that affect asymptotic relative efficiency (e.g., the shape of the loss function to minimize, the number of subvectors, the definition of cost) and the results of this Chapter do not imply that the cyclic and non-cyclic algorithms are equally efficient in general. For a theoretical analysis of asymptotic relative efficiency we refer the reader to Sections 6.2 and 6.3.

Chapter 8

A Zero-Communication Multi-Agent Problem

This chapter investigates the numerical performance of cyclic SA applied to a multi-agent optimization problem for tracking and surveillance with no communication between agents. Here, a group of agents equipped with finite-range sensors and no communication abilities can adjust their positions in order to collectively accomplish the following objectives:

Objective 1: Track any detected targets within a certain area.

Objective 2: Maximize the probability of detecting other targets in the area, that is maximize the collective coverage of the area of interest.

To address the two points above, it is assumed that each agent can obtain noisy

information regarding the current position of nearby agents and targets and, using this noisy information, each agent decides what its next action should be. The outline of this chapter is as follows. Section 8.1 describes the details behind the multi-agent optimization problem. Section 8.2 then proposes a cyclic SA approach for solving the optimization problem from Section 8.1. Lastly, Section 8.3 contains numerical results and Section 8.4 contains concluding remarks.

8.1 Problem Description

Throughout this chapter we let $\boldsymbol{x}_j(t) = [x_j^N(t), x_j^E(t), \dot{x}_j^N(t), \dot{x}_j^E(t)]^\top \in \mathbb{R}^4$ denote the vector of positions (in $\mathbb{R}^2$) and velocity (in $\mathbb{R}^2$) of agent j at time t. The four-dimensional $\boldsymbol{x}_j(t)$ is said to be the state vector of agent j at time t. Here, the superscripts "E" and "N" are used to denote agent j's position in the east and north directions, respectively, relative to some artificial center (corresponding to the point $(0, 0)$ on the Cartesian plane) that is universal to all agents and targets (this notation is used as a proxy for the standard (x, y) coordinate notation due to the fact that the letter "$\boldsymbol{y}$" will be used to denote the state vector of a target). The position and velocity of the ith target at time t will be represented by the state vector $\boldsymbol{y}_i(t) = [y_i^N(t), y_i^E(t), \dot{y}_i^N(t), \dot{y}_i^E(t)]^\top \in \mathbb{R}^4$ (the range of i is unspecified since the number of targets is unknown and may change with time). Each agent is assumed to be equipped with a sensor capa-

ble of estimating the positions of nearby agents and targets. The sensor model is described next.

8.1.1 The Sensor Model

When the target location is unknown, each agent is allowed to take a readings from its surroundings and collect estimated positions of any detected agents and targets. To model a sensor's probability of detection we use a variant of the sensor model from Kim et al. (2005). Here, it is assumed that there exists a constant $r > 0$, referred to as the range of the sensor, such that an agent cannot detect any target/agent located at a distance greater than r from the sensor. It is also assumed that there exists a constant $r' > 0$ with $r' < r$ such that any target/agent located within a distance of r' from the sensor's location will be detected (w.p.1). A target/agent located within the range of the sensor but at a distance larger than r' will be detected with a probability that decreases linearly as the distance to the target approaches r. Specifically, if d is the distance from a target/agent to a sensor, then the probability of detecting said target/agent is given by:

$$\text{Probability of Detection} = \begin{cases} 1 & \text{if } d \leq r', \\ 1 - \frac{d-r'}{r-r'} & \text{if } r' < d \leq r, \\ 0 & \text{if } d > r. \end{cases} \tag{8.1}$$

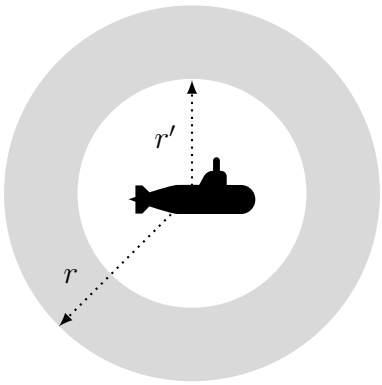


Figure 8.1: The probability of detection is 1 for agents/targets located within a radius of r'. The probability of detection in the gray annulus decreases as the distance to the agent's sensor increases. An agent cannot detect any agent/target located at a radius greater than r, the sensor's range.

Figure 8.1 illustrates (8.1).[1] The difference between the sensor model in sensor model in (8.1) and the sensor model in Kim et al. (2005) is that (8.1) assumes the probability of detection decreases linearly within the gay annulus in Figure 8.1, whereas Kim et al. (2005) allow this decrease to be nonlinear.

When an agent's sensor detects a target or another agent, we assume the sensor can also collect noisy information regarding the position of said target/agent. In the case where target i is detected by agent j at time t, then agent j can also obtain noisy measurements of

$$\varphi(\boldsymbol{x}_j(t), \boldsymbol{y}_i(t)) \equiv \tan^{-1}\left[\frac{y_i^N(t) - x_j^N(t)}{y_i^E(t) - x_j^E(t)}\right] + \chi\{y_i^E(t) - x_j^E(t) < 0\}\pi, \tag{8.2}$$

[1]The submarine graphic in Figure 8.1 was created by Freepik (www.freepik.com) and is licensed under the Attribution 3.0 Unported (CC BY 3.0) license.

the angle to the target (here $\chi\{\mathcal{E}\}$ is the indicator function of the event $\mathcal{E}$), and

$$\rho(\boldsymbol{x}_j(t), \boldsymbol{y}_i(t)) \equiv \sqrt{(y_i^N(t) - x_j^N(t))^2 + (y_i^E(t) - x_j^E(t))^2}, \tag{8.3}$$

the Euclidean distance to the target. The noisy measurements regarding the position of target i that are available to agent j are assumed to be of the form:

$$\boldsymbol{z}_{j:i}(t) \equiv \begin{bmatrix} \varphi(\boldsymbol{x}_j(t), \boldsymbol{y}_i(t)) \\ \\ \rho(\boldsymbol{x}_j(t), \boldsymbol{y}_i(t)) \end{bmatrix} + \boldsymbol{v} \in \mathbb{R}^2, \tag{8.4}$$

where $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ for a diagonal covariance matrix $\boldsymbol{R}$. A similar model to (8.4) is assumed for an agent detecting another agent. In the sensor model above, agents do now know the true position or velocity of the target(s) or of any other agent; only noisy observations of the distance and angle to any detected agent/target are available. However, it is assumed that an agent can correctly classify any detected individual as either an agent or a target. It is also assumed that an agent can temporarily assign names or "tags" to detected agents/targets order to match an agent/target's previous observed position to its current observed position. If an agent ever loses track (i.e., fails to detect) a previously detected agent/target, it is assumed that the tag for that agent/target resets, that is the agent can no longer match the previously detected agent/target to any agent/target detected in the future.

8.1.2 The Loss Function to Minimize

In order to motivate the choice of loss function to minimize, let us begin by considering a simplified setting where there is only one agent and one target. Here, if the agent detects the target at time t, a natural strategy for the agent to implement is to attempt to have $[x_1^E(t+\delta t), x_1^N(t+\delta t)]$, the agent's position at time $t+\delta t$ with $\delta t > 0$, resemble $[y_1^E(t+\delta t), y_1^N(t+\delta t)]$, the target's position at time $t+\delta t$. If $\boldsymbol{y}_1(t+\delta t)$ were known, the agent could attempt to minimize the function $\|[x_1^E(t+\delta t), x_1^N(t+\delta t)] - [y_1^E(t+\delta t), y_1^N(t+\delta t)]\|^2/2$ with respect to $[x_1^E(t+\delta t), x_1^N(t+\delta t)]$. Consider now the case where there are n agents and one target and assume agent j detects the target at time t. Furthermore, let us temporarily assume that agent j knows the values of $[y_1^E(t+\delta t), y_1^N(t+\delta t)]$ and $[x_i^E(t+\delta t), x_i^N(t+\delta t)]$ for $i \neq j$, that is agent j knows the future positions of all agents and targets (this assumption will be weakened later on). If agent j determines that several agents will be close to the target at time $t+\delta t$ then, in light of Objective 2 on p. 192, it may no longer be desirable for agent j to move towards the target's position. Instead, a more appropriate strategy for agent j might be to move away from the target's position in an attempt to maximize the overall probability of detecting other targets that may be in the area of interest (AOI). Even in this simplified setting where we are assuming that the future positions of the target and of all other agents are known to agent j, it is unclear *where specifically* agent j should move if it decides to move away from

the target's position. Motivated by the paper by Lee et al. (2015), we propose to let agent j move to the center of mass of its estimated Voronoi cell. This approach is based on the idea of attempting to have agents "spread out" when not actively tracking a target. Details are given next.

When agent j is not actively tracking a target, that is when the agent is not addressing Objective 1, the agent should be attempting to address Objective 2. In this light, agent j should take action to increase the overall coverage of the AOI (loosely speaking, we want the agents to "spread out" when not actively tracking a target). For this we define the following coverage function:

$$H(P_1, \ldots, P_n, [x_1^E, x_1^N]^\top, \ldots, [x_n^E, x_n^N]^\top) \equiv \sum_{j=1}^{n} \int_{P_j} \|\boldsymbol{q} - [x_j^E, x_j^N]^\top\|^2 \, d\boldsymbol{q}, \tag{8.5}$$

where $P_1, \ldots, P_n$ are subsets of $\mathbb{R}^2$ that form a partition of the AOI (a partition of the AOI is a collection of disjoint cells whose union is equal to the entire AOI). Minimizing (8.5) is equivalent to maximizing the coverage of the AOI. Moreover, it is known (see, for example, Lee et al. 2015) that minimizing (8.5) is equivalent to minimizing the following function:

$$F([x_1^E, x_1^N]^\top, \ldots, [x_n^E, x_n^N]^\top) \equiv \sum_{j=1}^{n} \int_{V_j} \|\boldsymbol{q} - [x_j^E, x_j^N]^\top\|^2 \, d\boldsymbol{q}, \tag{8.6}$$

where $V_j \subset \mathbb{R}^2$ is the jth Voronoi cell defined as the set of points that are at least as close (in Euclidean distance) to agent j than to any other agent (Figure

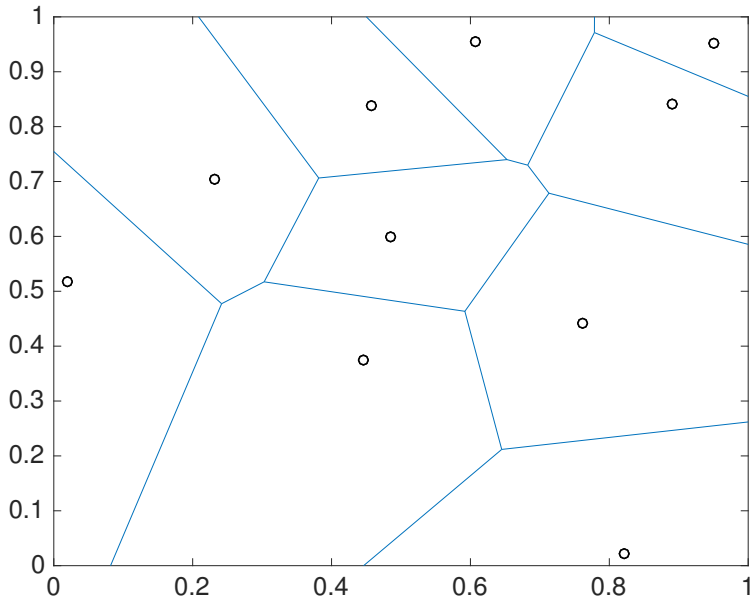


Figure 8.2: Voronoi tessellation where circles mark the positions of the agents. The Voronoi tessellation is unique. Cells depend on the positions of the agents.

8.2 gives an example of a Voronoi tessellation, a partition where the cells are Voronoi cells). While minimizing (8.6) is difficult, a critical point of (8.6) is obtained when $[x_j^E, x_j^N]^\top = \boldsymbol{c}_j$, where

$$\boldsymbol{c}_j = [c_j^E, c_j^N]^\top \equiv \frac{\int_{\boldsymbol{q} \in V_j} \boldsymbol{q} \; d\boldsymbol{q}}{\text{Area of } V_j}$$

is the center of mass of V_j. This suggests that a possible strategy for agent j to implement is to move towards $\boldsymbol{c}_j$ when not actively tracking a target.

So far, the discussion in this section suggests that if $[y_1^E(t+\delta t), y_1^N(t+\delta t)]$ and $\{[x_i^E(t+\delta t), x_i^N(t+\delta t)]\}_{i \neq j}$ are known to agent j at time t (a generally unreasonable assumption), then agent j should adjust its vector of velocities to

attempt to minimize the function:

$$
\begin{aligned}
&L_j(\dot{x}_j^E(t), \dot{x}_j^N(t)) \equiv \\
&\chi\{\gamma_1(t+\delta t) < \tau\} \left[\frac{\left\| [x_j^E(t+\delta t), x_j^N(t+\delta t)] - [y_1^E(t+\delta t), y_1^E(t+\delta t)] \right\|^2}{2} \right] \\
&+ \chi\{\gamma_1(t+\delta t) \geq \tau\} \left[\frac{\left\| [x_j^E(t+\delta t), x_j^N(t+\delta t)] - \boldsymbol{c}_j(t)^\top \right\|^2}{2} \right],
\end{aligned} \tag{8.7}
$$

where $\gamma_1(t+\delta t)$ is the number of agents (not counting agent j) that will be close to target 1 at time $t+\delta t$ (the definition of "close" is flexible), τ is a predetermined nonnegative integer, $\boldsymbol{c}_j(t)$ is the center of mass of the Voronoi cell, V_j, computed based on $\{[x_i^E(t+\delta t), x_i^N(t+\delta t)]\}_{i \neq j}$ and $[x_j^E(t), x_j^N(t)]$, and

$$
\begin{bmatrix} x_j^E(t+\delta t) \\ x_j^N(t+\delta t) \end{bmatrix} = \begin{bmatrix} x_j^E(t) \\ x_j^N(t) \end{bmatrix} + \delta t \begin{bmatrix} \dot{x}_j^E(t) \\ \dot{x}_j^N(t) \end{bmatrix}. \tag{8.8}
$$

The function:

$$
\begin{aligned}
L_j(\dot{x}_j^E(t), \dot{x}_j^N(t)) \equiv &\left[\sum_{i=1}^{\text{\# Targets}} \chi\{i = i^*\} \chi\{\gamma_i(t+\delta t) < \tau\} \times \right. \\
&\left. \frac{\left\| [x_j^E(t+\delta t), x_j^N(t+\delta t)] - [y_i^E(t+\delta t), y_i^N(t+\delta t)] \right\|^2}{2} \right] \\
&+ \left[\chi\{\gamma_i(t+\delta t) \geq \tau \text{ for all } i \text{ or \# Targets} = 0\} \times \right. \\
&\left. \frac{\left\| [x_j^E(t+\delta t), x_j^N(t+\delta t)] - \boldsymbol{c}_j(t)^\top \right\|^2}{2} \right]
\end{aligned} \tag{8.9}
$$

provides a natural generalization to (8.7) for the case where there may be zero or multiple targets; in (8.9), $\gamma_i(t+\delta t)$ is the number of agents (other than agent j) that will be close to target i at time $t+\delta t$, and $i^* \in S \equiv \{i$ such that $\gamma_i(t+\delta t) < \tau\}$ is such that out of all $i \in S$, i^* is the label/name of the target whose future position (at time $t+\delta t$) is closest to agent j's current position (at time t). If multiple targets in the set S are to be equally close to agent j at time $t+\delta t$, select one of these targets at random and let the selected target be target i^*. The indicator functions in (8.9) indicate whether agent j believes target i is already sufficiently tracked/covered by other agents.

Let us comment on the loss function from (8.9). First, note that this loss function is time-varying and possibly random since it depends on the time-varying position of the target and of all other agents. Second, $L_j(\dot{x}_j^E(t), \dot{x}_j^N(t))$ is agent-specific in that each agent is minimizing a different function (this can be seen by noting the presence of $\boldsymbol{c}_j(t)$ in the loss function, a term that is agent-dependent). Another thing to note is that although we have used the number of agents near the target as a measure of how well the target's location is covered, other measurements of target coverage could be used such as the probability that the target will be detected at time $t+\delta t$. The last observation we make pertains to the fact that the values of $[y_1^E(t+\delta t), y_1^N(t+\delta t)]$ and $\{[x_i^E(t+\delta t), x_i^N(t+\delta t)]\}_{i\neq j}$ are unknown to agent j at time t, as are the velocities of all other agents and targets. Therefore, the value of the function (8.9) is unknown to agent j

(recall that the variable $\gamma_i(t+\delta t)$ appearing in (8.9) depends on the state vectors of the other agents). The following section uses Kalman Filter-based estimates of agent- and target positions to approximate (8.9).

Using estimated values of the other agents' positions introduces an important issue. Suppose that agent j were to predict that $\gamma_i(t + \delta t)$ other agents would be located near target i at time $t + \delta t$ where $\gamma_i(t + \delta t) \geq \tau$. Then,the decision of agent j will be to move towards the center of mass of its Voronoi cell which could result in the agent moving away from the target. Note, however, that if the other $n - 1$ agents were to also predict that at least τ agents would be located near the target at time $t + \delta t$, then all n agents would move towards the centers of their Voronoi cells and away from the target, which could result in a problematic situation where the target is left "unguarded". To avoid this problem, we consider a setting in which agents update their velocity vectors in a strictly cyclic manner. As a result, when the time comes for agent j to update its velocity, the other agents will have already begun moving towards- or away from the target, giving agent j information about their intentions.

8.2 Proposed Cyclic SA-Based Approach

This section describes the precise way in which an agent estimates the future positions of detected agents and targets and how the agent uses this infor-

mation to adjust its state vector based on the loss function in (8.9). Specifically, this section answers the following questions:

1. How does agent j estimate $\{\boldsymbol{x}_i(t+\delta t)\}_{i\neq j}$ and $\boldsymbol{y}_i(t+\delta t)$ (i.e., the future state vectors of other agents and the state vectors of targets)?

2. How does agent j use the estimates of $\{\boldsymbol{x}_i(t+\delta t)\}_{i\neq j}$ and $\boldsymbol{y}_i(t+\delta t)$ to update its own state vector (position and velocity)?

Section 8.2.1 addresses the first question and Section 8.2.2 addresses the second question.

8.2.1 Extended Kalman Filter for Estimating State Vectors

At time t, agent j uses the extended Kalman filter (EKF) to obtain an estimate for $\boldsymbol{y}_i(t+\delta t)$, the true state vector of target i at time $t+\delta t$. The EKF is used due to the fact that (8.4) is nonlinear in the state vector. To use the EKF, agent j first models the dynamics of target i as follows:

$$\boldsymbol{y}_i(t+\delta t) = \boldsymbol{\Phi}\boldsymbol{y}_i(t) + \boldsymbol{w}(t) \tag{8.10}$$

for t in a discrete set of points δt time units apart, where

$$\boldsymbol{\Phi} \equiv \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.11}$$

and where $\boldsymbol{w}$ is a vector of zero-mean Gaussian white noise with covariance matrix $\boldsymbol{Q}$. The EKF algorithm used by agent j for estimating the target's state vector is illustrated in Algorithm 6. Loosely speaking, at time $t - \delta t$ agent j predicts the position of target i at time t (Line 1 of Algorithm 6). The predicted state estimate is denoted by $\hat{\boldsymbol{y}}_i(t|t-\delta t)$. Then, at time t, agent j uses the latest noisy information on the position of target i to correct its previous estimate (Line 6 of Algorithm 6) . The corrected state estimate is denoted by $\hat{\boldsymbol{y}}_i(t|t)$ (although $\boldsymbol{H}$, $\boldsymbol{K}$, and η are functions of time, this dependance has been omitted for simplicity). We assume that agent j also estimates the state vector of another detected agent, say agent i, using Algorithm 6 after a natural modification to replace $\hat{\boldsymbol{y}}_i(t|t)$ and $\hat{\boldsymbol{y}}_i(t|t-\delta t)$ with $\hat{\boldsymbol{x}}_i(t|t)$ and $\hat{\boldsymbol{x}}_i(t|t-\delta t)$, respectively.

One important thing to note is that the EKF algorithm defines $\boldsymbol{H}$ (appearing in Line 4 of Algorithm 6) as the Jacobian matrix of the vector:

$$\begin{bmatrix} \varphi(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t-\delta t)) \\ \rho(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t-\delta t)) \end{bmatrix} \tag{8.12}$$

with respect to $\boldsymbol{x}_j(t)$. However, the vector in (8.12) is not differentiable if and only if at least one of the following holds:

1. $x_j^E(t) = \hat{y}_j^E(t|t-\delta t)$ and $x_j^N(t) > \hat{y}_j^N(t|t-\delta t)$, where the vector $[\hat{y}_j^E(t|t-\delta t), \hat{y}_j^N(t|t-\delta t)]^\top$ corresponds to the first two entries of the vector $\hat{\boldsymbol{y}}_i(t|t-\delta t)$.

2. $\rho(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t-\delta t)) = 0$.

Algorithm 6 The Extended Kalman Filter (EKF) for Estimating $\boldsymbol{y}_i(t)$

Require: $\hat{\boldsymbol{y}}_i(t-\delta t|t-\delta t) \in \mathbb{R}^4$ and $\boldsymbol{P}_{t-\delta t|t-\delta t} \in \mathbb{R}^{4\times 4}$.

Predict (execute at time $t-\delta t$):

1: $\hat{\boldsymbol{y}}_i(t|t-\delta t) = \boldsymbol{\Phi}\hat{\boldsymbol{y}}_i(t-\delta t|t-\delta t)$, where $\boldsymbol{\Phi}$ is defined in (8.11).

2: $\boldsymbol{P}_{t|t-\delta t} = \boldsymbol{\Phi}\boldsymbol{P}_{t-\delta t|t-\delta t}\boldsymbol{\Phi}^\top + \boldsymbol{Q}$, where $\boldsymbol{Q}$ is defined below (8.11).

Correct (execute at time t):

3: $\boldsymbol{\eta} = \boldsymbol{z}_{j:i}(t) - [\varphi(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t-\delta t)), \rho(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t-\delta t))]^\top$, where $\varphi(\cdot,\cdot)$ and $\rho(\cdot,\cdot)$ are defined in (8.2) and (8.3).

4: $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}_{t|t-\delta t}\boldsymbol{H}^\top + \boldsymbol{R}$, where $\boldsymbol{R}$ is defined below (8.4) and:

$$\boldsymbol{H} \equiv \begin{bmatrix} \dfrac{x^N - y^N}{\rho^2(\boldsymbol{x},\boldsymbol{y})} & \dfrac{-(x^E - y^E)}{\rho^2(\boldsymbol{x},\boldsymbol{y})} & 0 & 0 \\ \dfrac{-(x^E - y^E)}{\rho(\boldsymbol{x},\boldsymbol{y})} & \dfrac{-(x^N - y^N)}{\rho(\boldsymbol{x},\boldsymbol{y})} & 0 & 0 \end{bmatrix} \text{ with } \boldsymbol{x} = \boldsymbol{x}_j(t) \text{ and } \boldsymbol{y} = \hat{\boldsymbol{y}}_i(t|t-\delta t).$$

5: $\boldsymbol{K} = \boldsymbol{P}_{t|t-\delta t}\boldsymbol{H}^\top\boldsymbol{S}^{-1}$.

6: $\hat{\boldsymbol{y}}_i(t|t) = \hat{\boldsymbol{y}}_i(t|t-\delta t) + \boldsymbol{K}\boldsymbol{\eta}$.

7: $\boldsymbol{P}_{t|t} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\boldsymbol{P}_{t|t-\delta t}$.

In other words, a successful implementation of the EKF algorithm requires avoiding scenarios 1 and 2 above, possibly by perturbing $\hat{\boldsymbol{y}}_i(t|t-\delta t)$.

8.2.2 Algorithm Description

We consider a setting in which the n agents update their directional velocities in a strictly cyclic manner. At time t_0, agent 1 uses its sensor to detect any nearby agents and targets. Agent 1 then estimates the future state vectors (at time $t_0 + \delta t$) of all detected agents and targets using the EKF algorithm

(Algorithm 6). Next, using the estimated future states of detected agents and targets, agent 1 updates its velocity vector (more details about this step will be given in the sequel). In the same manner, agent 2 updates its velocity vector at time $t_0 + \delta t/n$. In general, agent j updates its directional velocity at the times $\{t_0 + (j-1)\delta t/n + k\delta t\}_{k\geq 0}$ using the predicted positions of agents and targets δt time units into the future. Before implementing the algorithm, it is necessary to specify exactly how agent j updates its state vector. This is done next.

The objective of Agent j is to minimize the loss function in (8.9). As discussed in Section 8.1.2, however, agent j does not have access to a closed form expression for the loss function because the function depends on the unknown future state vectors of targets and of other agents. In our proposed approach, agent j uses the predicted agent- and target state vectors to estimate the gradient of the loss function (the differentiability of the loss function is addressed in Section 8.2.3). Agent j then updates its state vector using the noisy gradient estimate in a steepest-descent manner. Specifically, agent j updates its state vector at times $t \in \{t_0 + (j-1)\delta t/n + k\delta t\}_{k\geq 0}$ by following the steps below.

Step 1: Use sensor to detect nearby agents and targets.

Step 2: Estimate the future (at time $t + \delta t$) state vectors of all detected agents and targets using the EKF (Algorithm 6).

Step 3: Let $\lambda \geq 0$ be a number such that an agent is considered to be close

to the target if the distance between the agent and the target is less than λ (n general $\lambda \neq r$). For each detected target, estimate $\gamma_i(t+\delta t)$ (defined below (8.9)), based on λ and denote the estimate by $\hat{\gamma}_i(t+\delta t)$.

Step 4: Find the set of targets for which $\hat{\gamma}_i(t+\delta t) < \tau$ (with $\tau > 0$). Denote this set by $\hat{S}$ ($\hat{S}$ is therefore an estimate of the set S defined on p. 201).

Step 5: If $\hat{S} = \emptyset$ (here $\emptyset$ denotes the empty set), use the estimated future positions (at time $t+\delta t$) of the other agents along with the current position (at time t) of agent j to estimate $\boldsymbol{c}_j(t)$. Denote the estimate by $\hat{\boldsymbol{c}}_j(t)$. If $S \neq \emptyset$, pick the target in $\hat{S}$ whose predicted position is closest to agent j (in the event of a tie, choose a target at random). Denote that target "target $\hat{i}^*$" ($\hat{i}^*$ is an estimate for the integer i^* defined on p. 201).

Step 6: Select $a > 0$ and update agent j's velocity vector as follows:

$$
\begin{aligned}
&[\dot{x}_j^E(t), \dot{x}_j^N(t)]^\top = (1-a)[\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)]^\top \\
&+ a\Bigg[\chi\{\hat{\gamma}_i(t+\delta t) \geq \tau \text{ for all } i \text{ or \# Detected Targets} = 0\}\times \\
&\frac{\hat{\boldsymbol{c}}_j(t) - [x_j^E(t), x_j^N(t)]^\top}{\delta t}\Bigg] + \sum_{i=1}^{\text{\# Detected Targets}} a\Bigg[\chi\{\hat{\gamma}_i(t+\delta t) < \tau\}\chi\{i = \hat{i}^*\}\times \\
&\frac{[\hat{y}_i^E(t+\delta t|t), \hat{y}_i^N(t+\delta t|t)]^\top - [x_j^E(t), x_j^N(t)]^\top}{\delta t}\Bigg].
\end{aligned}
\tag{8.13}
$$

Step 7: Update agent j's position using (8.8) and (8.13).

8.2.3 Connection to Cyclic SA

First, note that the loss function in (8.9) is differentiable with gradient:

$$\frac{\partial L_j(\dot{x}_j^E, \dot{x}_j^N)}{\partial [\dot{x}_j^E, \dot{x}_j^N])^\top} = \Bigg[\chi\{\gamma_i(t+\delta t) \geq \tau \text{ for all } i \text{ or \# Targets} = 0\}\delta t \times \Big([x_j^E, x_j^N]^\top + \delta t [\dot{x}_j^E, \dot{x}_j^N]^\top - \boldsymbol{c}_j(t)^\top \Big) \Bigg] + \sum_{i=1}^{\text{\# Targets}} \Bigg[\chi\{\gamma_i(t+\delta t) < \tau\} \chi\{i = i^*\}\delta t \times \Big([x_j^E, x_j^N]^\top + \delta t [\dot{x}_j^E, \dot{x}_j^N]^\top - [y_i^E(t+\delta t), y_i^N(t+\delta t)]^\top \Big) \Bigg], \quad (8.14)$$

(assuming the future positions of the target do not depend on the positions of the agents, a simplifying consequence of (8.10)). The update in (8.13) is based on the idea that the gradient in (8.14) can be estimated using the values of $\hat{\boldsymbol{x}}_i(t+\delta t|t)$, and $\hat{\boldsymbol{y}}_i(t+\delta t|t)$. Specifically, define the stochastic gradient:

$$\hat{\boldsymbol{g}}_j(\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)) \equiv \Bigg[\chi\{\hat{\gamma}_i(t+\delta t) \geq \tau \text{ for all } i \text{ or \# Detected Targets} = 0\}\delta t \times \Big([x_j^E(t), x_j^N(t)] + \delta t [\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)] - \hat{\boldsymbol{c}}_j(t)^\top \Big) \Bigg]^\top + \sum_{i=1}^{\text{\# Detected Targets}} \Bigg[\chi\{\hat{\gamma}_i(t+\delta t) < \tau\} \chi\{i = \hat{i}^*\}\delta t \times \Big([x_j^E(t), x_j^N(t)] + \delta t [\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)] - [\hat{y}_i^E(t+\delta t|t), \hat{y}_i^N(t+\delta t|t)] \Big) \Bigg]^\top. \quad (8.15)$$

The vector in (8.15) can be seen as a stochastic estimate of the gradient in (8.14). Moreover, the update in (8.13) can be rewritten as:

$$[\dot{x}_j^E(t), \dot{x}_j^N(t)]^\top = [\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)]^\top - a(\delta t)^{-2}\hat{\boldsymbol{g}}_j(\dot{x}_j^E(t-\delta t), \dot{x}_j^N(t-\delta t)).$$

Thus, the update in (8.13) has the general form of the SA update in (2.4) with $a_k = a(\delta t)^{-2}$. A constant gain sequence was used due to the time-varying nature of the loss function. One important thing to note is that the noisy gradient estimate in (8.15) is not an unbiased estimate of the true gradient in (8.14). The bias is due to the fact that the EKF does not generally produce unbiased predictions of the future positions of any detected agents and targets. The magnitude of the bias will increase or decrease as the bias in the EKF estimates increases or decreases, respectively.

Because agents take turns performing the update in (8.13), the resulting algorithm resembles a distributed implementation of Algorithm 3, a cyclic algorithm which is a special case of the GCSA algorithm (Algorithm 1). However, the multi-agent algorithm from this chapter does not fit perfectly into the framework of Algorithm 1 due to the fact that each agent is optimizing a different loss function and each of these loss functions is time-varying. As a result, the theory from Chapters 4–6 is not directly applicable. The following section investigates the *numerical* performance of the algorithm in Section 8.2.2.

8.3 Numerical Analysis

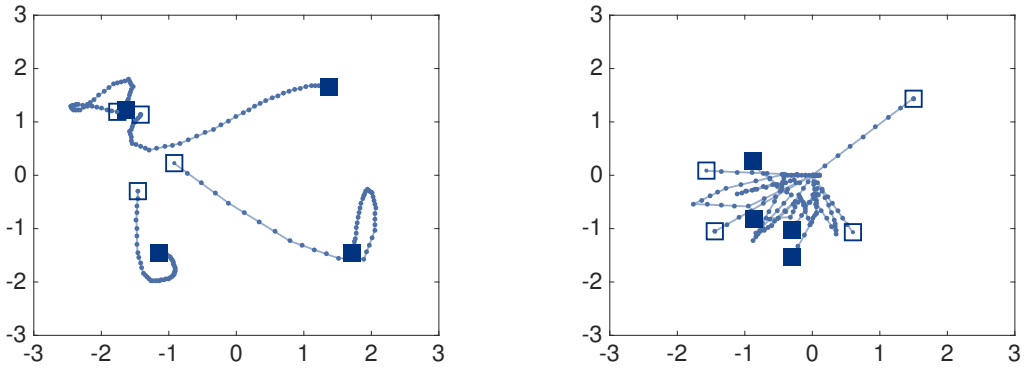
This section investigates the numerical performance of the algorithm from Section 8.2.2. The numerical results assume that the AOI is always a 6×6 square centered at the origin and that the value of a in (8.13) is such that $0 < a < 1$. The reason for forcing a to be strictly less than 1 stems from the fact that setting $a = 1$ implies that the distance measure $\rho(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t - \delta t)) = 0$, which implies that the matrix $\boldsymbol{H}$ (see Line 4 of Algorithm 6) as well as the angle to the target's predicted position are not well-defined. Having $a < 1$ guarantees that $\rho(\boldsymbol{x}_j(t), \hat{\boldsymbol{y}}_i(t|t - \delta t)) \neq 0$, avoiding the aforementioned complications.

8.3.1 Software

The numerical results in this Section are all performed in MATLAB. The MATLAB Multi-Parametric Toolbox 3.0 (Herceg et all. 2013) was used to compute the vertices defining the Voronoi cells, based on the current positions of the n agents, while simultaneously constraining the cells to be contained within the AOI. Then, the centers of mass of the resulting Voronoi cells (i.e., the $\hat{\boldsymbol{c}}_j(t)$s) were computed using the function `polygeom` by Sommer (2016).

8.3.2 Effect of the Sensor Range

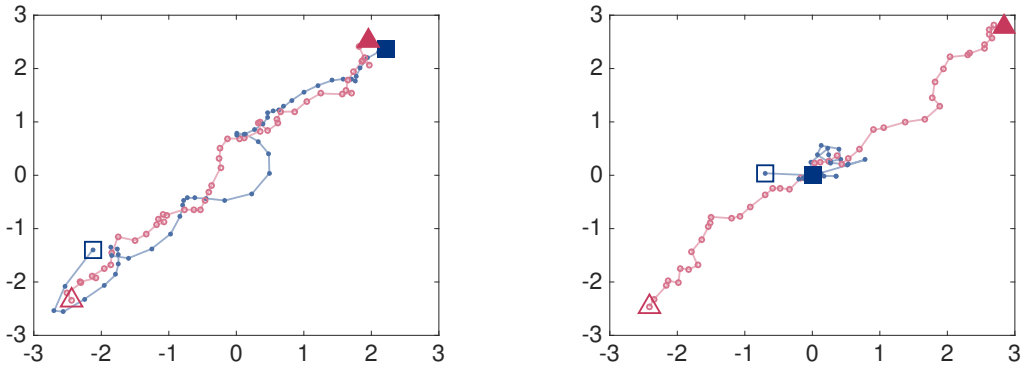
Because the algorithm in Section 8.2.2 involves zero communication between agents, the quality of the agents' sensors directly affects the performance



(a) $r = 4$, $r' = 3.8$. **(b)** $r = 0.5$, $r' = 0.4$.

Figure 8.3: The effect of the sensor range on area coverage for the case where there are 4 agents and 0 targets. Each empty square denotes the starting position of an agent. Similarly, each filled square denotes the final position of an agent. Note that in Figure 8.3b agents fail to spread out.

of the algorithm. Figure 8.3 presents two cases illustrating the effect of sensor range on area coverage for a setting where there are four agents and zero targets (both Figures 8.3a and 8.3b are the result of the agents updated their velocity 50 times). The first case, presented in Figure 8.3a, shows the evolution of four agents' positions when the radii $r = 2$ and $r' = 1.5$ (see (8.1) for the definitions of r and r'). Here, the four agents are frequently able to detect each other and can therefore spread out to maximize area coverage (i.e., minimize (8.6)). In contrast, Figure 8.3b shows the evolution of four agents' positions when $r = 0.5$ and $r' = 0.4$. Here, the agents start relatively far apart and fail to detect each other due to the limited range of the sensors. Consequently, the agents tend to gravitate towards the center of the AOI, an action that is not optimal when attempting to maximizing area coverage. In summary, Figure



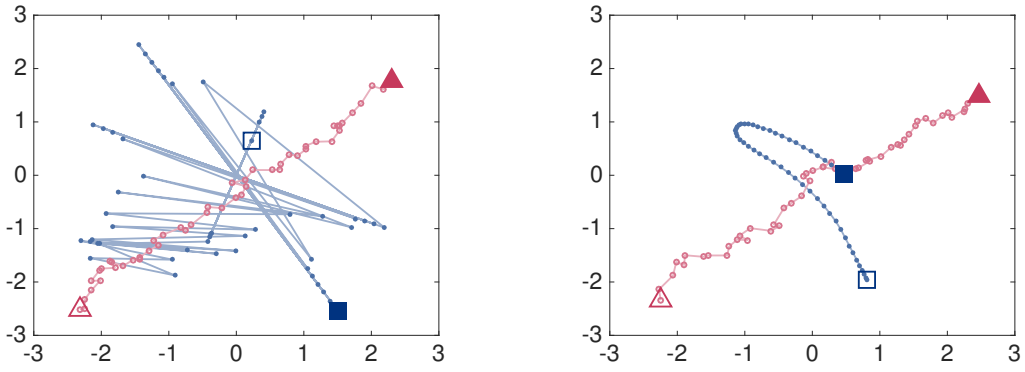
(a) $r = 2$, $r' = 1.5$.

(b) $r = 0.5$, $r' = 0.4$.

Figure 8.4: The effect of sensor range on target tracking for the case where there is 1 agent and 1 target. An empty square (respectively empty triangle) denotes the starting position of the agent (respectively target). The filled square (respectively filled triangle) denotes the final position of the agent (respectively target). Note that in Figure 8.4b the agent fails to track the target.

8.3 illustrates the fact that a larger sensor range provides agents with more of the information they need to distribute their positions optimally relative to the objective function in (8.6). Figure 8.3 was produced using $\boldsymbol{R} = 0.05\boldsymbol{I}$, $\delta t = 0.4$, and $a = 0.5$.

Aside from improving area coverage, a larger sensor range has the (not surprising) benefit of also improving target tracking. In Figure 8.4a, for example, a single agent whose sensor has a large range is able to track a single target fairly well. When the sensor's range is significantly decreased, however, Figure 8.4b shows that the agent is no longer able to continuously detect and keep track of the target. The simulations in Figure 8.4 were produced using $\boldsymbol{R} = 0.05\boldsymbol{I}$, $\delta t = 0.1$, and $a = 0.5$; and the target was assumed to move according to the linear model in (8.10) with $\boldsymbol{Q} = 0.01\boldsymbol{I}$ and velocity equal to 1 (i.e.,



(a) $a = 1.3$. **(b)** $a = 0.01$.

Figure 8.5: The effect of the gain sequence on target tracking for the case where there is 1 agent and 1 target. An empty square (respectively empty triangle) denotes the starting position of the agent (respectively target). The filled square (respectively filled triangle) denotes the final position of the agent (respectively target).

the last two entries of $\boldsymbol{y}_i(t)$ are equal to 1). Note that Figure 8.3 uses $\delta t = 0.4$ while Figure 8.4 uses $\delta t = 0.1$. This was done with the objective of simulating a setting which the agents begin updating their parameter vectors at times that are 0.1 time units apart, which implies $\delta t = 0.n$ when there are n agents.

8.3.3 Effect of the Gain Sequence

Like the sensor range, the value of a (the gain appearing in Step 6 of the multi-agent algorithm) plays a crucial role in the performance of the algorithm. Figure 8.5a, for example, shows an agent attempting to track a target when the gain sequence is too large. Here, the agent's velocity vector is highly sensitive to small changes in the target's predicted position, this eventually results in the agent moving too far from the target to the point where the agent is no longer

able to detect it. Similarly, Figure 8.5b shows how using a gain sequence that is too small also results in the agent not being able to remain close enough the target. Consequently, the agent eventually loses track of the target. The simulations in Figure 8.5 were produced using $\boldsymbol{R} = 0.05\boldsymbol{I}$, $\delta t = 0.1$, $a = 0.5$, $r = 2$, and $r' = 1.5$; and the target was assumed to move according to the linear model in (8.10) with $\boldsymbol{Q} = 0.01\boldsymbol{I}$ and velocity equal to 1 (i.e., the last two entries of $\boldsymbol{y}_i(t)$ are equal to 1). The agent and target updated their velocities 50 times.

Unfortunately, there is no general way to determine the best value of a in practice. One observation regarding the update in (8.13) that can be helpful in selecting the value of a is the following: setting $a = 1$ implies that $[x_j^E(t + \delta t), x_j^N(t+\delta t)]^\top$, the position of agent j at time $t+\delta t$, will be exactly equal to one of $[\hat{y}_i^E(t + \delta t|t), \hat{y}_i^N(t + \delta t|t)]^\top$ and $\hat{\boldsymbol{c}}_j(t)$, depending on whether the agent decides to move towards the target or towards the center of mass of its Voronoi cell. Therefore, when an agent is actively tracking a target, setting $a = 1$ implies that $\rho(\boldsymbol{x}_j(t + \delta t), \hat{\boldsymbol{y}}_i(t + \delta t|t)) = 0$. This observation implies that setting $a = 1$ is not an ideal strategy for two reasons:

1. The EKF algorithm is not well-defined when $\rho(\boldsymbol{x}_j(t + \delta t), \hat{\boldsymbol{y}}_i(t + \delta t|t)) = 0$ (see the discussion following (8.12)).

2. In reality, two bodies cannot occupy the same space so that having $\rho(\boldsymbol{x}_j(t+\delta t), \hat{\boldsymbol{y}}_i(t + \delta t|t)) = 0$ is not physically possible when $\hat{\boldsymbol{y}}_i(t + \delta t|t)$ is close to the target's true position at time $t + \delta t$.

In light of the discussion above, a general guideline for choosing a is to pick a value that is only slightly less than 1 if the target's predicted state is expected to be somewhat accurate. For our numerical experiments we select values of a that are strictly less than one in an attempt to ensure that the algorithm is not too sensitive to changes in the EKF's predictions.

8.3.4 Maximum Spread and Minimum Distance to Target

Sections 8.3.2 and 8.3.3 explained the significance of r, r', and a when a single agent is attempting to track a target. It was observed that if the sensor range is too small or if the value of a is either too small or too large, the terminal distance between the agent and the target tends to be large. When there are multiple agents in the AOI, two other variables that govern the algorithm's performance are λ, the distance at which a target is assumed to be close to an agent (see Step 3 of the algorithm), and τ, the number of agents that need to be near a target for that target to be considered to be well-guarded (see Step 4 of the algorithm). A large value of τ combined with a small value of λ implies that an agent will only move away from a detected target if it predicts that several agents will be extremely close to the target. In contrast, a small value of τ combined with a large value of λ implies that agents very easily move away from a detected target. Thus, a poor choice of τ and λ could result in either all agents moving away from the target or all agents gathering near the target;

neither of these scenarios is ideal given objectives 1 and 2 on p. 192. With this in mind, Tables 8.1–8.4 compute: 1) the mean terminal distance between the target and the nearest agent divided by the initial value of this distance (see the column "Dist. to Nearest Agent"), and 2) the mean terminal maximum distance between agents divided by the initial value of this distance (see the column "Max. Dist. Between Agents"), for different combinations of r, r', τ, and λ when there are 4 agents. Small values in the first column indicate agents are tracking the target while large values of the second column indicate agents are spreading out. The means in Tables 8.1–8.4 are computed from 20 i.i.d. replications, each consisting of 50 velocity updates per-agent. Each replication was produced by initializing the positions of agents and targets uniformly at random within the AOI; using $\boldsymbol{R} = 0.05\boldsymbol{I}$, $\delta t = 0.1$, and $a = 0.5$; and the target was assumed to move according to the linear model in (8.10) with $\boldsymbol{Q} = 0.01\boldsymbol{I}$, velocity equal to 1 (i.e., the last two entries of $\boldsymbol{y}_i(t)$ are equal to 1).

An overall trend that can be observed in Tables 8.1–8.4 is that larger values of r are correlated with a decrease in the mean terminal distance between the target and the nearest agent. Additionally, larger values of r are associated with larger values of the mean maximum distance between agents. In contrast, small values of r are correlated with a large mean terminal distance between the target and the nearest agent and a small mean terminal maximum distance between agents. Larger values of r allow agents to minimize

τ	λ	Dist. to Nearest Agent	Max. Dist. Between Agents
1	0.1	0.1569 ($\hat{\sigma}^2 = 0.0124$)	0.9537 ($\hat{\sigma}^2 = 0.1967$)
1	0.5	0.3348 ($\hat{\sigma}^2 = 0.1029$)	1.1006 ($\hat{\sigma}^2 = 0.3250$)
1	2	0.3048 ($\hat{\sigma}^2 = 0.0546$)	1.4699 ($\hat{\sigma}^2 = 0.1375$)
1	3	0.4875 ($\hat{\sigma}^2 = 0.1947$)	1.4973 ($\hat{\sigma}^2 = 0.5830$)
2	0.1	0.2964 ($\hat{\sigma}^2 = 0.0592$)	0.7556 ($\hat{\sigma}^2 = 0.1740$)
2	0.5	0.1976 ($\hat{\sigma}^2 = 0.0265$)	1.1087 ($\hat{\sigma}^2 = 0.4836$)
2	2	0.3080 ($\hat{\sigma}^2 = 0.0471$)	1.2940 ($\hat{\sigma}^2 = 0.3126$)
2	3	0.2400 ($\hat{\sigma}^2 = 0.0308$)	1.5787 ($\hat{\sigma}^2 = 0.6409$)
3	0.1	0.1949 ($\hat{\sigma}^2 = 0.0191$)	0.7750 ($\hat{\sigma}^2 = 0.2952$)
3	0.5	0.2330 ($\hat{\sigma}^2 = 0.0648$)	1.1551 ($\hat{\sigma}^2 = 1.0617$)
3	2	0.2703 ($\hat{\sigma}^2 = 0.0347$)	1.1701 ($\hat{\sigma}^2 = 0.3006$)
3	3	0.3261 ($\hat{\sigma}^2 = 0.1056$)	1.0590 ($\hat{\sigma}^2 = 0.1192$)

Table 8.1: Mean terminal distance from the target to the nearest agent and mean terminal maximum distance between agents when there are four agents, one target, $r = 3$, and $r' = 2.5$ (distances are divided by their values at the beginning of the replication prior to averaging). $\hat{\sigma}^2$ is the sample variance. The variables τ and λ have a significant effect over the behavior of the agents.

τ	λ	Dist. to Nearest Agent	Max. Dist. Between Agents
1	0.1	0.3557 ($\hat{\sigma}^2 = 0.0860$)	0.9675 ($\hat{\sigma}^2 = 0.2911$)
1	0.5	0.3261 ($\hat{\sigma}^2 = 0.0597$)	1.2445 ($\hat{\sigma}^2 = 0.2934$)
1	2	0.6423 ($\hat{\sigma}^2 = 0.3227$)	1.0706 ($\hat{\sigma}^2 = 0.1149$)
1	3	0.7195 ($\hat{\sigma}^2 = 0.3410$)	0.9462 ($\hat{\sigma}^2 = 0.0800$)
2	0.1	0.3513 ($\hat{\sigma}^2 = 0.1028$)	1.1958 ($\hat{\sigma}^2 = 0.3893$)
2	0.5	0.2993 ($\hat{\sigma}^2 = 0.0726$)	1.2100 ($\hat{\sigma}^2 = 0.1166$)
2	2	0.3120 ($\hat{\sigma}^2 = 0.0771$)	1.3101 ($\hat{\sigma}^2 = 0.2050$)
2	3	0.5354 ($\hat{\sigma}^2 = 0.3261$)	1.3315 ($\hat{\sigma}^2 = 0.4669$)
3	0.1	0.3448 ($\hat{\sigma}^2 = 0.3936$)	1.3020 ($\hat{\sigma}^2 = 0.3143$)
3	0.5	0.2593 ($\hat{\sigma}^2 = 0.0673$)	0.8850 ($\hat{\sigma}^2 = 0.1520$)
3	2	0.2373 ($\hat{\sigma}^2 = 0.0310$)	1.2419 ($\hat{\sigma}^2 = 0.1599$)
3	3	0.3159 ($\hat{\sigma}^2 = 0.1042$)	1.2420 ($\hat{\sigma}^2 = 0.1427$)

Table 8.2: Mean terminal distance from the target to the nearest agent and mean terminal maximum distance between agents when there are four agents, one target, $r = 2$, and $r' = 1.5$ (distances are divided by their values at the beginning of the replication prior to averaging). $\hat{\sigma}^2$ is the sample variance. The variables τ and λ have a significant effect over the behavior of the agents.

τ	λ	Dist. to Nearest Agent	Max. Dist. Between Agents
1	0.1	1.0161 ($\hat{\sigma}^2 = 0.6779$)	0.7132 ($\hat{\sigma}^2 = 0.1144$)
1	0.5	1.2612 ($\hat{\sigma}^2 = 0.7190$)	0.6005 ($\hat{\sigma}^2 = 0.1207$)
1	2	1.1480 ($\hat{\sigma}^2 = 0.8477$)	0.6960 ($\hat{\sigma}^2 = 0.1133$)
1	3	1.0841 ($\hat{\sigma}^2 = 0.3906$)	0.6978 ($\hat{\sigma}^2 = 0.2923$)
2	0.1	0.8769 ($\hat{\sigma}^2 = 0.4911$)	0.7907 ($\hat{\sigma}^2 = 0.1680$)
2	0.5	1.3092 ($\hat{\sigma}^2 = 0.8296$)	0.6923 ($\hat{\sigma}^2 = 0.1428$)
2	2	1.2736 ($\hat{\sigma}^2 = 0.3907$)	0.6780 ($\hat{\sigma}^2 = 0.1095$)
2	3	1.2974 ($\hat{\sigma}^2 = 1.2430$)	0.6500 ($\hat{\sigma}^2 = 0.0818$)
3	0.1	0.9303 ($\hat{\sigma}^2 = 0.6797$)	0.5529 ($\hat{\sigma}^2 = 0.0540$)
3	0.5	0.9133 ($\hat{\sigma}^2 = 0.4332$)	0.6480 ($\hat{\sigma}^2 = 0.1503$)
3	2	1.0092 ($\hat{\sigma}^2 = 0.4750$)	0.6979 ($\hat{\sigma}^2 = 0.0554$)
3	3	1.0755 ($\hat{\sigma}^2 = 0.5873$)	0.6411 ($\hat{\sigma}^2 = 0.0706$)

Table 8.3: Mean terminal distance from the target to the nearest agent and mean terminal maximum distance between agents when there are four agents, one target, $r = 1$, and $r' = 0.5$ (distances are divided by their values at the beginning of the replication prior to averaging). $\hat{\sigma}^2$ is the sample variance. The variables τ and λ have a significant effect over the behavior of the agents.

τ	λ	Dist. to Nearest Agent	Max. Dist. Between Agents
1	0.1	1.4262 ($\hat{\sigma}^2 = 0.5683$)	0.4505 ($\hat{\sigma}^2 = 0.0398$)
1	0.5	1.0955 ($\hat{\sigma}^2 = 0.3439$)	0.4335 ($\hat{\sigma}^2 = 0.0288$)
1	2	1.4668 ($\hat{\sigma}^2 = 0.1747$)	0.4351 ($\hat{\sigma}^2 = 0.0359$)
1	3	1.3079 ($\hat{\sigma}^2 = 0.5449$)	0.4665 ($\hat{\sigma}^2 = 0.0306$)
2	0.1	1.3213 ($\hat{\sigma}^2 = 0.5002$)	0.4293 ($\hat{\sigma}^2 = 0.0227$)
2	0.5	1.4793 ($\hat{\sigma}^2 = 0.9726$)	0.3979 ($\hat{\sigma}^2 = 0.0312$)
2	2	1.4770 ($\hat{\sigma}^2 = 0.3656$)	0.3648 ($\hat{\sigma}^2 = 0.0234$)
2	3	1.2614 ($\hat{\sigma}^2 = 0.6410$)	0.4079 ($\hat{\sigma}^2 = 0.0158$)
3	0.1	1.2997 ($\hat{\sigma}^2 = 0.3756$)	0.4262 ($\hat{\sigma}^2 = 0.0330$)
3	0.5	1.4249 ($\hat{\sigma}^2 = 0.8190$)	0.4505 ($\hat{\sigma}^2 = 0.0338$)
3	2	1.0037 ($\hat{\sigma}^2 = 0.1506$)	0.4659 ($\hat{\sigma}^2 = 0.0946$)
3	3	1.4076 ($\hat{\sigma}^2 = 0.7358$)	0.4224 ($\hat{\sigma}^2 = 0.0145$)

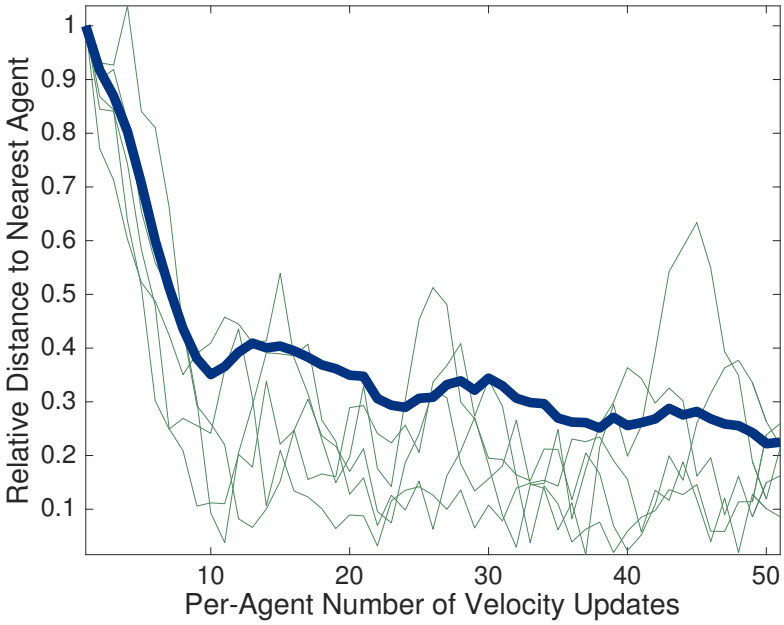
Table 8.4: Mean terminal distance from the target to the nearest agent and mean terminal maximum distance between agents when there are four agents, one target, $r = 0.5$, and $r' = 0.4$ (distances are divided by their values at the beginning of the replication prior to averaging). $\hat{\sigma}^2$ is the sample variance. The variables τ and λ have a significant effect over the behavior of the agents.

the distance between the target and the nearest agent while simultaneously maximizing the maximum distance between agents.

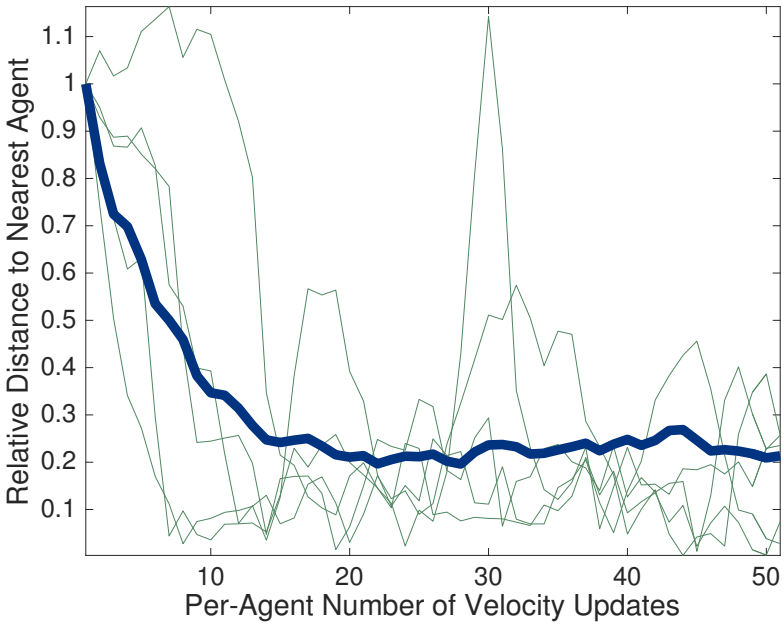
For the case where $r = 2$ and $r' = 1.5$, $\tau = 1$, and $\lambda = 0.5$, Figure 8.6a shows the evolution of the mean distance between the target and the nearest agent relative to the initial value of this distance using the settings from Tables 8.1–8.4. The result in Figure 8.6a is consistent with the results in Table 8.2. Figure 8.6b repeats the experiment of Figure 8.6a for the case where there is one agent and one target. The mean terminal distance in Figure 8.6b seems to be approximately the same as the mean terminal distance in Figure 8.6a, although it is important to note the high variance of the individual realizations

8.4 Concluding Remarks

This chapter investigates the numerical performance of cyclic SA applied to a multi-agent optimization problem for tracking and surveillance. An attractive feature of the resulting multi-agent optimization algorithm is that it can be implemented when agents are not allowed to communicate with each other, although it is entirely possible to extend the algorithm to include communication between agents. One assumption made throughout this section is that each agent knows their own position and velocity at any point in time. Through a natural modification, the algorithm in this chapter could also be



(a) Four agents, one target.



(b) One agent, one target.

Figure 8.6: Mean distance from the target to the nearest agent divided by the initial value of this distance. The thick line represents the mean relative distance obtained by averaging 20 i.i.d. realizations while each of the thinner lines represents one of the first 5 realizations.

implemented in the same manner when these variables are estimated. The numerical results indicate that, when the sensor radius is sufficiently large, the proposed algorithm allows agents to track the target while simultaneously increasing the distance between agents.

A few directions for future research regarding this chapter's multi-agent algorithm include investigating other methods for predicting target/agent state positions (aside from the EKF), considering adaptive gain sequences (instead of the using the constant gain a), and considering other objective functions. With regards to this last topic, we remind the reader that in this chapter each agent is minimizing a different loss function, which is a different setting than the setting from Chapters 3–7 where there is a universal (i.e., common to all agents) loss function to minimize. A universal loss function can be constructed for the multi-agent problem in this chapter by simply adding the individual loss functions. Specifically, the universal loss function could be defined as $L(\theta) \equiv \sum_{j=1}^{n} L_j(\dot{x}_j^E(t), \dot{x}_j^N(t))$, where θ is a vector containing the velocity vectors of all n agents. Note that the value of $[\dot{x}_j^E(t), \dot{x}_j^N(t)]^\top$ affects the value of $L_i(\dot{x}_i^E(t), \dot{x}_i^N(t))$ even when $i \neq j$ (through the terms $\boldsymbol{c}_i(t)$ and $\gamma_\ell(t + \delta t)$ for $\ell = 1, \ldots, \#\text{ Agents}$). Furthermore, if agent j is located at a distance of exactly λ from any target (under the usual Gaussian state model this occurs with probability zero) then $L(\theta)$ is not differentiable with respect to $[\dot{x}_j^E(t), \dot{x}_j^N(t)]^\top$. Future work could focus on studying this more complex scenario.

Chapter 9

Overall Concluding Remarks and Future Work

This dissertation investigated the asymptotic properties of the generalized cyclic stochastic approximation (GCSA) algorithm (Algorithm 1) including the convergence (w.p.1) of the iterates, the asymptotic normality of the normalized iterates, and the asymptotic efficiency of GCSA relative to its non-cyclic counterpart. Next we review some of the contributions of this dissertation.

The main theorem on convergence, Theorem 2, provided sufficient conditions for the convergence w.p.1 of the GCSA iterates to a zero of the gradient of the function to minimize. Although some of the convergence conditions closely resemble well-known conditions for the convergence of the standard (i.e., non-cyclic) SA algorithms, a few conditions are inherent only to the GCSA

algorithm (see Section 4.5 for a discussion on the validity of the convergence conditions in Theorem 2). Section 7.1 contains numerical examples illustrating the convergence of GCSA using both SG- and SPSA-based gradient estimates.

With the goal of computing the rate of convergence of the GCSA algorithm, a generalization to Theorem 2.2 in Fabian (1968) (regarding the asymptotic normality of SA procedures) was provided in Chapter 5 (see Theorem 4). Theorem 5 then used the generalization from Theorem 4 to derive conditions for the asymptotic normality of the normalized iterates of a special case of GCSA (which helps define the rate of convergence of the algorithm) as well as to obtain expressions for the parameters of its asymptotic distribution (see Section 5.5 for a discussion on the conditions of Theorem 5 for asymptotic normality). A few other applications of our generalization to Theorem 2.2 in Fabian (1968) are discussed in Appendix A, including an application to an adaptive SA algorithm. Numerical examples supporting the theory on asymptotic normality are given in Section 7.2.

Using the result on asymptotic normality from Theorem 5, Chapter 6 provided an analytical estimate for the asymptotic efficiency (in the MSE sense) of a special case of GCSA relative to the efficiency of its non-cyclic counterpart after taking into consideration the cost of implementing each algorithm. It was shown that, in general, either algorithm may be (asymptotically) more efficient than the other (it is important to keep in mind, however, that the rel-

ative efficiency in Chapter 6 is only one possible way to compare the cyclic and non-cyclic algorithms). Section 7.3 contains numerical experiments estimating the aforementioned relative efficiency under different definitions of cost when the update directions are either SG- or SPSA-based.

While the numerical examples in Chapter 7 focused on cases where the conditions for convergence and/or asymptotic normality were met, Chapter 8 investigated the numerical performance of a cyclic SA algorithm applied to a zero-communication multi-agent optimization problem where the loss function is time-varying. As a result, the theory of Chapters 4–6 (which assumes that the loss function is not time-varying) does not apply. It was observed that the performance of the cyclic SA approach had a strong correlation with the quality of the agents' sensors: the better the sensors' quality, the better the cyclic algorithm performed (this behavior is not surprising and, more importantly, is not specific to the cyclic algorithm).

We end this chapter by giving a few topics for future research. One natural direction for future research pertains to the convergence of a constrained variant of GCSA. A possible approach to constructing a constrained version of GCSA could involve projecting $\hat{\theta}_k$ at every step (we refer the reader to the comment on p. 90 regarding condition A4). Another approach could consist on constructing a variant of GCSA for which the constraints are satisfied asymptotically (e.g., Wang and Spall 2011, where the iterates of the algorithm asymp-

totically satisfy the requirement of lying in a discrete set). The work of Wang and Spall (2008) presents an SA algorithm based on the penalty functions method for solving stochastic optimization problems with general inequality constraints, the ideas of this paper may also be of interest when considering constrained versions of the GCSA algorithm.

A second avenue for future research involves the generalization of the GCSA algorithm to an asynchronous setting (i.e., parallelizing GCSA), this could possibly be done using ideas similar to those in Tsitsiklis (1994). Additionally, it would also be interesting to study the behavior of cyclic procedures for time-varying loss functions (many multi-agent optimization problems, such as the problem in Chapter 8, have a time-varying loss function).

A few other topics for future research are the following:

1. Can the boundedness assumption of condition A1 (see p. 57) be weakened? (Answering this question may prove useful in when obtaining practical conditions for the convergence of an asynchronous variant of GCSA.)

2. Borkar and Meyn (2000) show that:

   > . . . the ODE method can be extended to establish both the stability and convergence of the stochastic approximation method, as opposed to only the latter. (Borkar and Meyn 2000)

   Might the ideas of the paper by Borkar and Meyn (2000) be used to extend the theory of convergence of the GCSA iterates in an analogous manner?

3. Could iterate averaging be combined with Algorithm 2 (see p. 78) so that the iterates of the resulting algorithm are asymptotically normally distributed (after an appropriate centering and scaling)?

4. What finite-time performance guarantees can be obtained for the GCSA algorithm (e.g., in terms of bounds on the MSE as a function of the iteration number)?

5. As discussed in Sections 6.1 and 6.2, the cost of implementation is an important aspect to consider when comparing a cyclic algorithm to its non-cyclic counterpart. Future work might consist of a more detailed investigation into how the cost of implementing a cyclic algorithm compares to the cost of implementing a non-cyclic algorithm.

In summary, this dissertation addressed several important (and previously unanswered) questions regarding cyclic implementations of SA procedures. Still, several interesting questions remain unanswered (as indicated in the discussion above). It is expected that some of the directions for future work mentioned above will make heavy use of the results and ideas in this dissertation.

Appendix A

A Generalization of Fabian (1968) and Applications

Stochastic approximation (SA) is a general framework for analyzing the convergence of a large collection of stochastic root-finding algorithms. The Kiefer–Wolfowitz and stochastic gradient algorithms are two well known (and widely used) examples of SA. Because of their applicability to a wide range of problems, many results have been obtained regarding the convergence properties of SA procedures. One important reference in the literature, Fabian (1968), derives general conditions for the asymptotic normality of the SA iterates. Since then, many results regarding the asymptotic normality of SA procedures have relied heavily on Theorem 2.2 in Fabian (1968) (which we refer to as "Fabian's theorem"). Unfortunately, some of the assumptions of Fabian's theorem are not

applicable to some modern implementations of SA in control and learning (Section 5.2, for example, explained why Fabian's theorem could not be applied to the GCSA algorithm). This chapter explains in detail the nature of this incompatibility and shows how Fabian's theorem can be generalized to address the issue.[1] While the main theorem in this chapter has already been presented in Chapter 5 (the main result in this chapter is the same as Theorem 4 in Chapter 5), the theorem was previously presented in the context of cyclic SA. This chapter discusses other applications of the theorem for a more general class of SA algorithms (due to an effort to make this chapter somewhat self-contained, there is some overlap between this chapter and Chapter 5). Furthermore, this appendix contains a proof of the generalization to Fabian's theorem that is more detailed than the proof provided in Section 5.3.

The main result in Fabian (1968), which we will refer to as "Fabian's theorem," is concerned with the following recursion:

$$\boldsymbol{W}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\boldsymbol{\Gamma}_k)\boldsymbol{W}_k + \frac{\boldsymbol{T}_k}{k^{\alpha+\beta/2}} + \frac{\boldsymbol{\Phi}_k \boldsymbol{V}_k}{k^{(\alpha+\beta)/2}}, \tag{A.1}$$

for $\alpha, \beta > 0$, random random vectors $\boldsymbol{W}_k, \boldsymbol{T}_k, \boldsymbol{V}_k \in \mathbb{R}^p$, and random matrices $\boldsymbol{\Phi}_k, \boldsymbol{\Gamma}_k \in \mathbb{R}^{p \times p}$. Under certain conditions, Fabian (1968) shows that $k^{\beta/2}\boldsymbol{W}_k$ is asymptotically normally distributed (with mean vector and covariance matrix determined by α, β, $\boldsymbol{T}_k$, $\boldsymbol{V}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{\Gamma}_k$. Undoubtedly, Fabian's theorem is

[1] This chapter is largely based on the paper by Hernandez and Spall (2017).

already applicable to a variety of SA algorithms (see, for example, Zhou and Hu 2014, Kar et al. 2013, Hu et al. 2012, and Zorin et al. 2000 to name a few applications). However, a critical assumption in Fabian's theorem is that $\Gamma_k \to \Gamma$ with probability one (w.p.1) for some real, positive definite matrix Γ. This introduces an important limitation, the nature of which we review next.

Let us begin by describing the standard SA recursion. Given a vector $\theta \in \mathbb{R}^p$ and a vector-valued function $\boldsymbol{f}(\theta) \in \mathbb{R}^n$ the basic SA algorithm for solving $\boldsymbol{f}(\theta) = 0$ is given by the recursion:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \boldsymbol{Y}_k(\hat{\theta}_k), \tag{A.2}$$

where $\boldsymbol{Y}_k(\theta)$ is a vector-valued random variable representing a noisy observation of $\boldsymbol{f}(\hat{\theta}_k)$, and $a_k > 0$ with $a_k \to 0$ is the gain sequence of the algorithm. Under certain conditions (which vary depending on the specific SA algorithm considered) we have $\hat{\theta}_k \to \theta^*$ w.p.1, where θ^* is a zero of $\boldsymbol{f}(\theta)$. Letting $\boldsymbol{W}_k = \hat{\theta}_k - \theta^*$ in (A.1), Fabian's theorem can be used to derive conditions under which $k^{\beta/2}(\hat{\theta}_k - \theta^*)$ has a limiting multivariate normal distribution. For some SA algorithms, however, requiring Γ to be *symmetric* (which is necessary in order for Γ to be real and positive definite) is incompatible with Fabian's other assumptions regarding α, β, $\boldsymbol{T}_k$, $\boldsymbol{V}_k$, and $\boldsymbol{\Phi}_k$. Specifically, for some SA algorithms it impossible to find a parametrization of (A.1) such that Γ is sym-

metric and all of Fabian's assumptions on α, β, $\boldsymbol{T}_k$, $\boldsymbol{V}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{\Gamma}_k$ also hold. Consequently, Fabian's theorem cannot be applied. The following are examples of such algorithms:

1. SA algorithms for root-finding where $\boldsymbol{f}(\theta)$ does not represent a gradient so that the Jacobian of $\boldsymbol{f}(\theta)$ is typically non-symmetric (e.g., Blum 1954).

2. Second order or adaptive SA algorithms for stochastic optimization where $\boldsymbol{f}(\theta)$ is the gradient of a function to minimize and a_k is replaced by a diagonal matrix that is not necessarily a multiple of the identity matrix (e.g., Renotte and Wouwer 2003).

3. The Generalized Cyclic SA (GCSA) algorithm in which only a subset of the parameter vector is updated at any given time. The subvector to update can be selected following a deterministic pattern (Hernandez and Spall 2014 and 2016) or according to a random variable (Hernandez 2016).

4. Randomized coordinate descent algorithms where some coordinates may be updated with a higher probability than others (e.g., Nesterov 2010).

The main contribution of this work is to provide a theorem that relaxes Fabian's restriction that $\boldsymbol{\Gamma}$ must be symmetric. The generalization makes the theorem more applicable to some modern applications of SA in control and learning including special cases of examples 1–4 above.

It is important to note that there do exist results showing the asymptotic normality of certain classes of SA procedures for which Γ may not be a symmetric matrix. In general, however, explicit expressions for the parameters of the asymptotic distribution are only available for specific SA algorithms. For the special case where $a_k = 1/k$ with $k \geq 1$, for example, Nevel'son and Has'minskii (1973, Chapter 6) show asymptotic normality for the Robbins–Monro procedure (Robbins and Monro 1951) where Γ is not required to be symmetric and is allowed to have complex eigenvalues. The authors give a closed-form expression for the mean vector and covariance matrix of the asymptotic distribution.

One common class of results closely related to asymptotic normality are the stochastic differential equation (SDE)-based definitions of rate of convergence for SA algorithms (e.g., Chapter 10 in Kushner and Yin 1997, Chapter 4 in part II of Benveniste et al. 1990, and Chapter 7 in Kushner and Clark 1978). Here, the authors first normalize the SA iterates and rewrite the resulting normalized iterates as a continuous time-dependent processes. The authors then show that the normalized continuous process converges weakly to an SDE and the rate of convergence is defined as the inverse of the scaling coefficient when normalizing the SA iterates. In certain cases, such as when the resulting SDE corresponds to a stationary Gauss–Markov process, these results imply the asymptotic normality of the normalized SA iterates and Γ is not required to be symmetric. Generally, however, the connection between asymptotic normal-

ity and the SDE-based definition of rate of convergence is not explicitly made. Moreover, in the instances where a connection to asymptotic normality is made, the parameters of the asymptotic distribution are not typically provided. Benveniste et al. (1990, Part II, Section 4.5, Theorem 13), for example, use the SDE-based method to show asymptotic normality of normalized SA iterates where Γ is not required to be symmetric and may have complex eigenvalues. However, to find the covariance matrix of the asymptotic distribution one must solve the Lyapunov equation on p. 334 of the same reference. We also note that Benveniste's result is limited to the case where the mean of the asymptotic distribution is zero (e.g., does not include the algorithms from Section 2.4).

A few attractive features of our proposed generalization are that 1) the result is not restricted to any specific SA algorithm, 2) aside from guaranteeing asymptotic normality, we also present the explicit solution for the asymptotic mean and covariance matrix that is not generally provided in the SDE-based approach above, and 3) our generalization makes Fabian's theorem applicable to some SA algorithms for which asymptotic normality has not been proven.

A.1 Preliminaries

While we refer the reader to Theorem 3 of this dissertation for the full set of conditions of Fabian's theorem, a few key assumptions are that Γ_k must con-

verge to a real positive definite (and therefore symmetric) matrix $\boldsymbol{\Gamma}$ w.p.1, $\boldsymbol{T}_k$ must converge to some vector $\boldsymbol{T}$ either w.p.1 (with probability one) or in expectation, $\boldsymbol{\Phi}_k$ must converge to $\boldsymbol{\Phi}$ w.p.1, $\boldsymbol{V}_k$ must have mean zero (conditionally on $\mathcal{F}_k$), and the covariance matrix of $\boldsymbol{V}_k$ (conditional on $\mathcal{F}_k$) must be uniformly bounded over k and $\mathcal{F}_k$ and must converge (w.p.1) to some positive definite matrix $\boldsymbol{\Sigma}$. When $\boldsymbol{f}(\theta) = \boldsymbol{g}(\theta)$ is the gradient of a function to minimize, say $L(\theta)$, the assumption that $\boldsymbol{\Gamma} = \lim_{k\to\infty} \boldsymbol{\Gamma}_k$ may be reasonable. To see this, note that if $\boldsymbol{g}(\theta)$ is continuously differentiable then (A.2) may be written in the form of (A.1) by letting $\boldsymbol{\Gamma}_k = k^{\alpha} a_k \tilde{\boldsymbol{H}}_k$, where the ith row of $\tilde{\boldsymbol{H}}_k$ is equal to the ith row of the Hessian matrix, $\boldsymbol{H}(\theta)$, of $L(\theta)$ evaluated at $\theta = (1-\lambda_i)\hat{\theta}_k + \lambda_i \theta^*$ for some $\lambda_i \in [0,1]$ that depends on $\hat{\theta}_k$. If $\boldsymbol{H}(\theta)$ is continuous at θ^*, $\hat{\theta}_k \to \theta^*$ w.p.1, and $k^{\alpha} a_k \to a > 0$, then requiring $\boldsymbol{\Gamma}$ to be real and positive definite translates into $a\boldsymbol{H}(\theta^*)$ being real and positive definite, a common assumption in many minimization problems. Suppose now that a modification is made to (A.1) in which $\boldsymbol{f}(\theta) = \boldsymbol{g}(\theta)$ and a_k is replaced by a diagonal matrix $\boldsymbol{A}_k$ (e.g., Renotte and Wouwer 2003 where $\boldsymbol{A}_k$ is taken to be a diagonal matrix with the same diagonal entries as an estimate of $a_k \boldsymbol{H}(\hat{\theta}_k)^{-1}$). Here, since $\boldsymbol{Y}_k(\hat{\theta}_k)$ denotes a noisy estimate of the gradient it is common to replace the notation $\boldsymbol{Y}_k(\hat{\theta}_k)$ in (A.2) with $\hat{\boldsymbol{g}}_k(\hat{\theta}_k)$. This modification gives rise to the algorithm:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - \boldsymbol{A}_k \hat{\boldsymbol{g}}_k(\hat{\theta}_k). \tag{A.3}$$

It is easy to verify that (A.3) is a special case of (A.1) in which $\boldsymbol{\Gamma}_k = k^{\alpha}\boldsymbol{A}_k\tilde{\boldsymbol{H}}_k$. Therefore, if $\boldsymbol{H}(\theta)$ is continuous, $\hat{\theta}_k \to \theta^*$ w.p.1, and $k^{\alpha}\boldsymbol{A}_k \to \boldsymbol{A}$ w.p.1 then $\boldsymbol{\Gamma}_k \to \boldsymbol{\Gamma} = \boldsymbol{A}\boldsymbol{H}(\theta^*)$ w.p.1. Generally the matrix $\boldsymbol{A}\boldsymbol{H}(\theta^*)$ would not be symmetric and, therefore, Fabian's theorem would not be applicable. Because there is no *unique* way to define the variables $\boldsymbol{\Gamma}_k$, $\boldsymbol{T}_k$, $\boldsymbol{\Phi}_k$, and $\boldsymbol{V}_k$ in (A.1), it may be tempting to think that it is always possible to redefine these variables so that $\boldsymbol{\Gamma}$ is symmetric. Next we show that such a redefinition often leads to very strong assumptions on $\hat{\theta}_k$.

Suppose an SA algorithm can be written in the form of (A.1) and that all of the conditions of Fabian's theorem are satisfied *with the exception that* $\boldsymbol{\Gamma}$ *is not symmetric*. For simplicity, we consider a special case of (A.1) where $\beta \neq 0$, $\boldsymbol{T}_k = \boldsymbol{T}$, $\boldsymbol{\Phi}_k = \boldsymbol{I}$, and $\boldsymbol{\Gamma}_k = \boldsymbol{\Gamma}$. Now, assume the matrices $\boldsymbol{\Gamma}'_k$, $\boldsymbol{\Phi}'_k$ and vectors $\boldsymbol{T}'_k$, $\boldsymbol{V}'_k$ provide an alternative way to write the SA algorithm in the form of (A.1) and assume $\boldsymbol{\Gamma}'_k$ converges w.p.1 to a symmetric matrix. Then, since

$$\boldsymbol{W}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\boldsymbol{\Gamma}'_k)\boldsymbol{W}_k + \frac{\boldsymbol{T}}{k^{\alpha+\beta/2}} + \frac{\boldsymbol{V}_k}{k^{(\alpha+\beta)/2}} + k^{-\alpha}(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k,$$

it is known that either $\boldsymbol{T}'_k$ must depend on $(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k$ or $\boldsymbol{\Phi}'_k\boldsymbol{V}'_k$ must depend on $(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k$. However, the assumptions on $\boldsymbol{T}'_k$, $\boldsymbol{\Phi}'_k$, and $\boldsymbol{V}'_k$ are typically incompatible with the term $k^{\beta/2}(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k$. Say, for example, that we let $\boldsymbol{\Phi}'_k = \boldsymbol{I}$, $\boldsymbol{V}'_k = \boldsymbol{V}_k$, and $\boldsymbol{T}'_k = \boldsymbol{T} + k^{\beta/2}(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k$. Here, having $\boldsymbol{T}'_k$ converge to

some finite vector $\boldsymbol{T}'$ w.p.1 or in expectation (as required by Fabian's theorem) would impose a priori conditions on the stochastic rate at which $\hat{\theta}_k$ converges to θ^*. However, such a condition violates the very purpose of Fabian's theorem, which is to establish such a rate of convergence. Alternatively, having $\boldsymbol{\Phi}'_k \boldsymbol{V}'_k = \boldsymbol{V}_k + k^{(\beta-\alpha)/2}(\boldsymbol{\Gamma}'_k - \boldsymbol{\Gamma})\boldsymbol{W}_k$ does not lead to an appropriate definition of $\boldsymbol{\Phi}'_k$ and $\boldsymbol{V}'_k$ given the restriction that $\boldsymbol{V}'_k$ must have mean zero conditionally on $\mathcal{F}_k$. By generalizing Fabian's conditions to relax the symmetry condition on $\boldsymbol{\Gamma}$ we avoid the need for imposing any additional restrictions on $\hat{\theta}_k$. Next we give a few other examples of algorithms for which $\boldsymbol{\Gamma}$ cannot be assumed to be symmetric.

Another algorithm for which $\boldsymbol{\Gamma}$ may not be symmetric is a variant of the adaptive stochastic approximation method for stochastic optimization in Spall (2000). Given a function $L(\theta)$ to minimize, the algorithm in Spall (2000) is as follows:

$$\begin{aligned}
\hat{\theta}_{k+1} &= \hat{\theta}_k - a_k \overline{\overline{\boldsymbol{H}}}_k^{-1} \hat{\boldsymbol{g}}_k(\hat{\theta}_k), \\
\overline{\boldsymbol{H}}_k &\equiv \frac{k}{k+1} + \overline{\boldsymbol{H}}_{k-1} + \frac{1}{k+1}\hat{\boldsymbol{H}}_k, \\
\overline{\overline{\boldsymbol{H}}}_k &\equiv \boldsymbol{\pi}_k(\overline{\boldsymbol{H}}_k),
\end{aligned} \tag{A.4}$$

where $\hat{\boldsymbol{H}}_k$ is an estimate of the Hessian of $L(\theta)$ evaluated at $\hat{\theta}_k$ and $\boldsymbol{\pi}_k(\overline{\boldsymbol{H}}_k)$ maps $\overline{\boldsymbol{H}}_k$ onto the set of positive definite matrices. When the dimension of $\overline{\boldsymbol{H}}_k$

is high, it may be advantageous to let $\overline{\overline{\boldsymbol{H}}}_k$ be a diagonal matrix. However, since $\boldsymbol{\Gamma}_k = k^{\alpha} a_k \overline{\overline{\boldsymbol{H}}}_k^{-1} \tilde{\boldsymbol{H}}_k$ for algorithm (A.4) (see Spall 2000, Theorems 3a and 3b) then the requirement that $\boldsymbol{\Gamma}$ must be symmetric generally prevents us from using Fabian's theorem when $\boldsymbol{\pi}_k$ is a mapping onto the set of diagonal matrices (here we note that the conditions for asymptotic normality given in Spall 2000, which are based on Fabian's theorem, require that $\overline{\overline{\boldsymbol{H}}}_k$ converge to $\boldsymbol{H}(\theta^*)$ precisely so that $\boldsymbol{\Gamma}$ is symmetric and this condition also prevents $\overline{\overline{\boldsymbol{H}}}_k$ from being a diagonal matrix unless $\boldsymbol{H}(\theta^*)$ is also a diagonal matrix). Requiring $\boldsymbol{\Gamma}$ to be symmetric is also an unreasonable assumption for the cyclic seesaw SA algorithm where $\boldsymbol{\Gamma}_k = k^{\alpha} \boldsymbol{A}_k \tilde{\boldsymbol{H}}_k$ as in the case of algorithm (A.3) (see (6) in Hernandez 2016). In cases where SA is used for general root-finding, the symmetry of $\boldsymbol{\Gamma}$ is an especially unreasonable assumption. Here, $\boldsymbol{\Gamma}_k = k^{\alpha} a_k \tilde{\boldsymbol{J}}_k$ where the ith row of $\tilde{\boldsymbol{J}}_k$ is equal to the ith row of the Jacobian of $\boldsymbol{f}(\theta)$ evaluated at $\theta = (1 - \lambda_i)\hat{\theta}_k + \lambda_i \theta^*$ for some $\lambda_i \in [0, 1]$.

In the proof of Fabian's theorem, the symmetry of $\boldsymbol{\Gamma}$ is used to write $\boldsymbol{\Gamma} = \boldsymbol{P} \boldsymbol{\Lambda} \boldsymbol{P}^{\top}$ for a real orthogonal matrix $\boldsymbol{P}$ and a real diagonal matrix $\boldsymbol{\Lambda}$ with strictly positive eigenvalues. Both $\boldsymbol{P}$ and $\boldsymbol{\Lambda}$ affect the parameters of the limiting distribution of $k^{\beta/2}(\hat{\theta}_k - \theta^*)$. When $\boldsymbol{\Gamma}$ is not a symmetric matrix, writing $\boldsymbol{\Gamma} = \boldsymbol{P} \boldsymbol{\Lambda} \boldsymbol{P}^{\top}$ is clearly not possible. This is a problem for all previous examples where $\boldsymbol{\Gamma}$ can reasonably be assumed to satisfy $\boldsymbol{\Gamma} = \boldsymbol{A}\boldsymbol{M}$ for a positive definite diagonal matrix $\boldsymbol{A}$ and a matrix $\boldsymbol{M}$ which represented either a Hessian or a Jacobian.

Although this Γ is not generally symmetric, it may still be real diagonalizable and have strictly positive eigenvalues (i.e., $\Gamma = \boldsymbol{S}\Lambda\boldsymbol{S}^{-1}$ for a nonsingular, not necessarily orthogonal, real matrix $\boldsymbol{S}$ and a positive definite diagonal matrix Λ). Such is the case, for example, if $\boldsymbol{M} = \boldsymbol{H}(\theta^*)$ is real and positive definite. For nonlinear root-finding SA algorithms, however, $\boldsymbol{M} = \boldsymbol{J}(\theta^*)$ represents the Jacobian of $\boldsymbol{f}(\theta)$ evaluated at θ^*. Here, it is possible for $\Gamma = \boldsymbol{A}\boldsymbol{J}(\theta^*)$ not to be diagonalizable even if $\boldsymbol{J}(\theta^*)$ is a square matrix. In the special case where Γ is a square matrix, however, $\Gamma = \boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}$ for a nonsingular real matrix $\boldsymbol{S}$ and a real upper triangular matrix $\boldsymbol{U}$ (here it is said that Γ is real upper-triangularizable) with strictly positive diagonal entries if and only if Γ is real and has strictly positive eigenvalues (see, for example, Horn and Johnson p. 82). Generalizing Fabian's result to allow Γ to be any real upper-triangularizable matrix with strictly-positive eigenvalues would then allow for treatment of the aforementioned "non-standard" algorithms, which is precisely the generalization we introduce here.

A.2 The Generalized Theorem

This section contains a generalization to Fabian's theorem derived by replacing Fabian's Assumption 2.2.1 with a weaker assumption. Following the proof of Fabian's theorem (Fabian 1968), we begin by showing that $k^{\beta/2}\boldsymbol{W}_k$ is

asymptotically normally distributed if and only if a much simpler process is also asymptotically normally distributed. After showing that the simpler process does, in fact, converge in distribution to a multivariate normal random variable, the parameters of its asymptotic distribution will uniquely determine the parameters of the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$.

Throughout this Chapter M_{ij} and $M_{k(i,j)}$ denote the (i,j)th entries of the matrices $\boldsymbol{M}$ and $\boldsymbol{M}_k$, respectively, and v_i and $v_{k(i)}$ denote the ith entries of the vectors $\boldsymbol{v}$ and $\boldsymbol{v}_k$, respectively. Furthermore, $k \geq 1$ denotes a strictly positive integer; $\xrightarrow{\text{dist}}$ means convergence in distribution; $\boldsymbol{V}_k$, $\boldsymbol{W}_k$, $\boldsymbol{T}_k$, and $\boldsymbol{T}$ are vectors in $\mathbb{R}^p$; $\boldsymbol{\Gamma}_k$, $\boldsymbol{\Phi}_k$, $\boldsymbol{\Sigma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Phi}$, and $\boldsymbol{P}$ are matrices in $\mathbb{R}^{p\times p}$; and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{M})$ denotes a multivariate normal random variable with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{M}$. Furthermore, throughout this section we assume the recursion for $\boldsymbol{W}_k$ given in (A.1) satisfies the following conditions:

B0′ $\boldsymbol{\Gamma}_k$, $\boldsymbol{\Phi}_{k-1}$, and $\boldsymbol{V}_{k-1}$ are $\mathcal{F}_k$-measurable where $\mathcal{F}_k$ is a non-decreasing sequence of σ-fields with $\mathcal{F}_k \subset \mathcal{S}$ for some σ-field $\mathcal{S}$.

B1′ There exists an upper triangular $\boldsymbol{U} \in \mathbb{R}^{p\times p}$ with strictly positive eigenvalues and a nonsingular $\boldsymbol{S} \in \mathbb{R}^{p\times p}$ such that $\boldsymbol{\Gamma}_k \to \boldsymbol{\Gamma} = \boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}$ w.p.1.

B2′ $\boldsymbol{\Phi}_k \to \boldsymbol{\Phi}$ w.p.1.

B3′ Either $\boldsymbol{T}_k \to \boldsymbol{T}$ w.p.1 or $E\|\boldsymbol{T}_k - \boldsymbol{T}\| \to 0$.

B4′ $E[\boldsymbol{V}_k|\mathcal{F}_k] = \boldsymbol{0}$, there exists a constant $C > 0$ such that $C > \|E[\boldsymbol{V}_k\boldsymbol{V}_k^\top|\mathcal{F}_k] -$

$\boldsymbol{\Sigma}\|$, and $\|E[\boldsymbol{V}_k\boldsymbol{V}_k^\top|\mathcal{F}_k] - \boldsymbol{\Sigma}\| \to 0$ w.p.1.

B5′ For some $0 < \alpha \leq 1$ define $\sigma_{k,r}^2 \equiv E\chi\{\|\boldsymbol{V}_k\|^2 \geq rk^\alpha\}\|\boldsymbol{V}_k\|^2$ where $\chi\{\mathcal{E}\}$ is the indicator function of the event $\mathcal{E}$. For every $r > 0$ either:

$$\lim_{k\to\infty} \sigma_{k,r}^2 = 0 \text{ or } \alpha = 1 \text{ and } \lim_{n\to\infty} n^{-1}\sum_{k=1}^{n} \sigma_{k,r}^2 = 0.$$

B6′ Define $\lambda \equiv \min_i \{U^{[i,i]}\}$. Let α and β be constants such that $0 < \alpha \leq 1$ and $0 \leq \beta$. Define $\beta_+ \equiv \beta$ if $\alpha = 1$ and $\beta_+ \equiv 0$ otherwise. Then, $\beta_+ < 2\lambda$.

Conditions B0′ and B2′–B6′ are identical to the conditions of Fabian's theorem (note that B6′ is obtained by rewriting B6 in the notation of B1′). On the other hand, B1′ is the relaxed version of Fabian's corresponding condition (condition 2.2.1 in Fabian 1968) requiring symmetry of $\boldsymbol{\Gamma}$. As discussed in the comment following Proposition 1, any real square matrix with real eigenvalues satisfies B1′. Next we use conditions B0′–B6′ to relate the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ to that of a much simpler process.

In order to compute the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ we begin by constructing a slightly different process, $\tilde{\boldsymbol{W}}_k$, defined as follows:

$$\tilde{\boldsymbol{W}}_k = (k-1)^{\beta/2}\boldsymbol{S}^{-1}\boldsymbol{W}_k, \tag{A.5}$$

where $\boldsymbol{S}$ is the matrix from B1′. It is easy to see that $\boldsymbol{W}_1 = \boldsymbol{0}$. Moreover, us-

ing Slutsky's theorem it follows that if $\tilde{\boldsymbol{W}}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{M})$ then $k^{\beta/2}\boldsymbol{W}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{S}\boldsymbol{\mu}, \boldsymbol{S}\boldsymbol{M}\boldsymbol{S}^\top)$. Thus, proving that the process $\tilde{\boldsymbol{W}}_k$ is asymptotically normally distributed (with certain mean vector and covariance matrix) is sufficient for computing the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$. Moreover, after some algebraic manipulation it can be shown that $\tilde{\boldsymbol{W}}_k$ is a special case of (A.1):

$$\tilde{\boldsymbol{W}}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{\Gamma}}_k)\tilde{\boldsymbol{W}}_k + k^{-\alpha}\boldsymbol{S}^{-1}\boldsymbol{T}_k + k^{-\alpha/2}\boldsymbol{S}^{-1}\boldsymbol{\Phi}_k\boldsymbol{V}_k, \tag{A.6}$$

where $\tilde{\boldsymbol{W}}_1 = \boldsymbol{0}$,

$$\tilde{\boldsymbol{\Gamma}}_k \equiv \left(\frac{k}{k-1}\right)^{\beta/2}\boldsymbol{S}^{-1}\boldsymbol{\Gamma}_k\boldsymbol{S} - \left[\left(\frac{k}{k-1}\right)^{\beta/2} - 1\right]k^{\alpha}\boldsymbol{I},$$

and $\tilde{\boldsymbol{\Gamma}}_k \to \tilde{\boldsymbol{U}} \equiv \boldsymbol{U} - (\beta_+/2)\boldsymbol{I}$ w.p.1. Using the same arguments as those in Fabian (1968, proof of Theorem 2.2) it can be shown that replacing $\boldsymbol{T}_k$ with $\boldsymbol{T}$ and $\boldsymbol{\Phi}_k$ with $\boldsymbol{\Phi}$ in (A.6) does not change the asymptotic distribution of $\tilde{\boldsymbol{W}}_k$ (note that the recursive nature of $\tilde{\boldsymbol{W}}_k$ means this is not immediate). Therefore, in order to show that $\tilde{\boldsymbol{W}}_k$ is asymptotically normally distributed we may assume, without loss of generality, that $\boldsymbol{T}_k = \boldsymbol{T}$ and $\boldsymbol{\Phi}_k = \boldsymbol{\Phi}$ so that:

$$\tilde{\boldsymbol{W}}_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{\Gamma}}_k)\tilde{\boldsymbol{W}}_k + k^{-\alpha}\tilde{\boldsymbol{T}} + k^{-\alpha/2}\tilde{\boldsymbol{V}}_k, \tag{A.7}$$

where $\tilde{\boldsymbol{T}} \equiv \boldsymbol{S}^{-1}\boldsymbol{T}$ and $\tilde{\boldsymbol{V}}_k \equiv \boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{V}_k$. The following lemma facilitates future

analysis by relating the asymptotic distribution of $\tilde{\boldsymbol{W}}_k$, as described by (A.7), to that of an even simpler process.

Lemma 14. Consider the following process:

$$\tilde{\boldsymbol{W}}'_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{\Gamma}}_k)\tilde{\boldsymbol{W}}'_k + k^{-\alpha/2}\tilde{\boldsymbol{V}}_k, \tag{A.8}$$

where $\tilde{\boldsymbol{W}}'_1 \equiv \boldsymbol{0}$. Assume the process $\boldsymbol{W}_k$ in (A.1) satisfies B0′–B6′. If $\tilde{\boldsymbol{W}}'_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{M})$ then $\tilde{\boldsymbol{W}}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\nu}, \boldsymbol{M})$, where

$$\boldsymbol{\nu}_i \equiv (\tilde{U}_{ii})^{-1}\tilde{T}_i - \left[(\tilde{U}_{ii})^{-1}\sum_{j=i+1}^{p}\tilde{U}_{ij}\boldsymbol{\nu}_j\right] \tag{A.9}$$

Proof. The result of the lemma holds if $\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k \to \boldsymbol{\nu}$ w.p.1. We show this next. First, using the triangle inequality it follows that

$$\|\tilde{\boldsymbol{W}}_{k+1} - \tilde{\boldsymbol{W}}'_{k+1}\| \leq (1 - k^{-\alpha}[\lambda - \beta_+/2 + o(1)])\|\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k\| + \|k^{-\alpha}\tilde{\boldsymbol{T}}\|, \tag{A.10}$$

where $o(\cdot)$ is the standard little-*o* notation. Then, (A.10) and Lemma 4.2 in Fabian (1967) imply $\limsup_{k\to\infty}\|\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k\| < \infty$ w.p.1 (we write $\|\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k\| = O(1)$ w.p.1 where $O(\cdot)$ is the standard big-*O* notation). Next, note that

$$\tilde{\boldsymbol{W}}_{k+1} - \tilde{\boldsymbol{W}}'_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{U}})(\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k) + k^{-\alpha}[\tilde{\boldsymbol{T}} + (\tilde{\boldsymbol{U}} - \tilde{\boldsymbol{\Gamma}}_k)(\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k)]. \tag{A.11}$$

Then, since $\|\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k\| = O(1)$ w.p.1 and $\tilde{\boldsymbol{\Gamma}}_k \to \tilde{\boldsymbol{U}}$ w.p.1 we may assume each entry of the term $(\tilde{\boldsymbol{U}} - \tilde{\boldsymbol{\Gamma}}_k)(\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k)$ is $o(1)$ w.p.1. Applying Lemma 4.2 (Fabian 1967) to the last entry of $\tilde{\boldsymbol{W}}_k - \tilde{\boldsymbol{W}}'_k$ as determined by recursion (A.11) implies:

$$\tilde{\boldsymbol{W}}_{k(p)} - \tilde{\boldsymbol{W}}'_{k(p)} \to \nu_p = (\tilde{U}_{pp})^{-1}\tilde{T}_p \text{ w.p.1}.$$

This is consistent with (A.9). In general, Lemma 4.2 in Fabian (1967) can be used to show that $\tilde{\boldsymbol{W}}_{k(i)} - \tilde{\boldsymbol{W}}'_{k(i)} \to \nu_i$ w.p.1 where ν_i is as in (A.9). Thus, ν_i can be computed once the values $\{\nu_\ell\}_{\ell=i+1}^{p}$ have been computed (i.e., once the last $p - i$ entries of $\boldsymbol{\nu}$ have been computed). □

Lemma 14 implies that deriving the asymptotic distribution of $\tilde{\boldsymbol{W}}'_k$ is sufficient for deriving the asymptotic distribution of $\tilde{\boldsymbol{W}}_k$ and, therefore, of $k^{\beta/2}\boldsymbol{W}_k$. At this point, the same arguments as those in Fabian (1968, proof of Theorem 2.2) can be used to show the following two results under B0′–B6′:

1. Replacing $\tilde{\boldsymbol{\Gamma}}_k$ with $\tilde{\boldsymbol{U}}$ in (A.8) does not change the asymptotic distribution of $\tilde{\boldsymbol{W}}'_k$ which depends on the limit (w.p.1) of $\tilde{\boldsymbol{\Gamma}}_k$ but not on $\tilde{\boldsymbol{\Gamma}}_k$ itself. Therefore, without loss of generality we may assume that $\tilde{\boldsymbol{\Gamma}}_k = \tilde{\boldsymbol{U}}$ so that:

$$\tilde{\boldsymbol{W}}'_{k+1} = (\boldsymbol{I} - k^{-\alpha}\tilde{\boldsymbol{U}})\tilde{\boldsymbol{W}}'_k + k^{-\alpha/2}\tilde{\boldsymbol{V}}_k. \tag{A.12}$$

2. The characteristic function of the asymptotic distribution of $\tilde{\boldsymbol{W}}'_k$ evaluated

at $\boldsymbol{t} \in \mathbb{R}^p$ depends on $\boldsymbol{t}$, α, $\tilde{\boldsymbol{U}}$, and on $\tilde{\boldsymbol{\Sigma}} \equiv \lim_{k\to\infty} \mathrm{var}[\tilde{\boldsymbol{V}}_k|\mathcal{F}_k]$ but does not depend on other aspects of the distribution of $\tilde{\boldsymbol{V}}_k$ (provided B0′–B6′ hold). Therefore, we may assume that the vectors $\tilde{\boldsymbol{V}}_k$ are i.i.d. $\mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Sigma}})$.

The following Lemma derives the asymptotic distribution of $\tilde{\boldsymbol{W}}'_k$.

Lemma 15. Assume $\boldsymbol{W}_k$ (defined in Theorem 3) satisfies B0′–B6′. Then, $\tilde{\boldsymbol{W}}'_k$ converges in distribution to a multivariate normal random variable with mean zero and covariance matrix $\boldsymbol{Q}$ whose entries are the unique solution to:

$$Q_{ij} = \frac{[\boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top(\boldsymbol{S}^{-1})^\top]_{ij}}{\tilde{U}_{ii}+\tilde{U}_{jj}} - \left[\frac{\sum_{\ell=j+1}^p \tilde{U}_{j\ell}\, Q_{i\ell} + \sum_{\ell=i+1}^p \tilde{U}_{i\ell}\, Q_{\ell j}}{\tilde{U}_{ii}+\tilde{U}_{jj}}\right]. \tag{A.13}$$

Proof. First, since $\tilde{\boldsymbol{V}}_k = \boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{V}_k$ then $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{S}^{-1}\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^\top(\boldsymbol{S}^{-1})^\top$ by condition B4′. Next, by the discussion immediately preceding this lemma we may assume (without loss of generality) that the vectors $\tilde{\boldsymbol{V}}_k$ are i.i.d. $\mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Sigma}})$. Consequently, for each k the distribution of the random vector $\tilde{\boldsymbol{W}}'_k$ from (A.12) must be a multivariate normal random variable with mean zero and covariance $\boldsymbol{Q}_k \equiv E[\tilde{\boldsymbol{W}}_k\tilde{\boldsymbol{W}}_k^\top]$. Using (A.12), the entries of $\boldsymbol{Q}_k$ can be shown to satisfy:

$$\begin{aligned} Q_{k+1(ij)} = {} & \left(1 - k^{-\alpha}\left[\tilde{U}_{ii} + \tilde{U}_{jj} - k^{-\alpha}\tilde{U}_{ii}\,\tilde{U}_{jj}\right]\right) Q_{k(ij)} \\ & - k^{-\alpha}\left[\sum_{\ell=j+1}^p \tilde{U}_{j\ell}\, Q_{k(i\ell)} + \sum_{\ell=i+1}^p \tilde{U}_{i\ell}\, Q_{k(\ell j)}\right] + k^{-\alpha}\tilde{\Sigma}_{ij} \\ & + k^{-2\alpha}\sum_{\ell=i}^p \tilde{U}_{i\ell} \sum_{s=j+1}^p \tilde{U}_{js}\, Q_{\ell s} + k^{-2\alpha}\sum_{\ell=i+1}^p \tilde{U}_{i\ell}\,\tilde{U}_{jj}\, Q_{\ell j}. \end{aligned} \tag{A.14}$$

Note that for the special case where $i = j = p$ we have:

$$Q_{k+1(pp)} = \left(1 - k^{-\alpha}\left[\tilde{U}_{pp} + \tilde{U}_{pp} - o(1)\right]\right) Q_{k(pp)} + k^{-\alpha}\tilde{\Sigma}_{pp}.$$

Then, Lemma 4.2 (Fabian 1967) implies that $Q_{pp} = \tilde{\Sigma}_{pp}/[\tilde{U}_{pp} + \tilde{U}_{pp}]$. More generally, assume Q_{mn} has already been computed for all tuples (m, n) such that either $m \geq i+1$ and $n \geq j+1$, $m = i$ and $n \geq j+1$, or $m \geq i+1$ and $n = j$. Then, (A.14) implies:

$$\begin{aligned} Q_{k+1(ij)} = (1 - k^{-\alpha}[\tilde{U}_{ii} + \tilde{U}_{jj} - o(1)])Q_{k(ij)} \\ - k^{-\alpha}\left[\sum_{\ell=j+1}^{p} \tilde{U}_{j\ell}\, Q_{k(i\ell)} + \sum_{\ell=i+1}^{p} \tilde{U}_{i\ell}\, Q_{k(\ell j)} - o(1) - \tilde{\Sigma}_{ij}\right]. \end{aligned}$$

Then, Lemma 4.2 in Fabian (1967) implies $\boldsymbol{Q} = \lim_{k\to\infty} \boldsymbol{Q}_k$ satisfies (A.13). Therefore, the characteristic function of $\boldsymbol{W}_k'$ converges to the characteristic function of a multivariate normal random variable with mean zero and covariance $\boldsymbol{Q}$. This implies the desired result. □

The following theorem uses the result of Lemmas 14 and 15 to compute the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$.

Theorem 6 (This is identical to Theorem 4)**.** Assume the recursion for $\boldsymbol{W}_k$ given in (A.1) satisfies conditions B0′–B6′. Then, the asymptotic distribution of $k^{\beta/2}\boldsymbol{W}_k$ is a multivariate normal random variable with mean $\boldsymbol{S}\boldsymbol{\nu}$ and covariance

matrix $\boldsymbol{SQS}^\top$, where the entries of $\boldsymbol{\nu}$ are the unique solution to (A.9) and the entries of $\boldsymbol{Q}$ are the unique solution to (A.13). Entry Q_{ij} can be computed once Q_{mn} has been computed for all tuples (m, n) such that either $m \geq i+1$ and $n \geq j+1$, $m = i$ and $n \geq j+1$, or $m \geq i+1$ and $n = j$. Therefore, the entries of $\boldsymbol{Q}$ can be computed beginning with Q_{pp} and using the symmetry of $\boldsymbol{Q}$. Similarly, the entries of $\boldsymbol{\nu}$ can be computed beginning with ν_p.

Proof. First, Lemma 15 shows $\tilde{\boldsymbol{W}}_k' \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$. Next, Lemma 14 implies that $\tilde{\boldsymbol{W}}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{\nu}, \boldsymbol{Q})$. Finally, by the discussion following (A.5) we see that $\boldsymbol{W}_k \xrightarrow{\text{dist}} \mathcal{N}(\boldsymbol{S\nu}, \boldsymbol{SQS}^\top)$ as desired. □

Note that if $\boldsymbol{\Gamma}$ is real and symmetric then the terms in square brackets in (A.9) and (A.13) disappear since $\tilde{\boldsymbol{U}}$ may be taken to be a diagonal matrix.

Appendix B

System Identification for Multi-Sensor Data Fusion

This chapter considers the problem of determining the presence and location of a static object within an area of interest (AOI) by combining information from multiple sensors.[1] A simple maximum-likelihood (ML) approach is investigated. We consider a setting in which there exist two main types of sensors, namely: "small" and "large" sensors. Essentially, it is assumed that small sensors can inspect an area that is relatively small in comparison to that which the large sensor can inspect. By deriving a relationship between small and large sensor measurements we combine data using the aforementioned ML-based approach. In particular, each detection problem is initially formulated

[1] This chapter is largely based on the paper by Hernandez (2015).

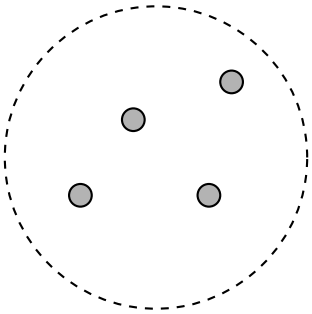


Figure B.1: The large sensor is capable of inspecting the entire area within the dotted region (the AOI) while each of the small sensors inspects one of the shaded regions. In the notation of this chapter, each of the shaded regions corresponds to a C_i. Without loss of generality we assume $p = 5$ with one cell left unexplored (the complement of the small sensor search areas within the AOI) which is denoted by C_5. Therefore, the AOI $= \cup_{i=1}^{5} C_i$.

as a system identification problem. Here, the large sensor collects data on the full system while small sensors collect data on subsystems. By establishing a connection of this identification problem to existing literature, we can obtain asymptotic convergence and asymptotic normality results. It is important to mention that the terms "object" and "AOI" will be used in a very general sense. Two examples will be considered. The first example is the search for a static object within a given area (e.g., Chung and Burdick 2012). Here, the terms object and AOI can be taken literally. The second example, however, is the detection of faulty tanks in a three-tank system (TTS) (e.g., Zhou et al. 2012). In this case, detecting a faulty tank within the TTS can also be thought of as detecting an object within an AOI.

An important assumption in our proposed approach is that there are two classes of sensors: a large sensor S capable of searching the entire AOI for

an object and a set of small sensors $\{S_1, \ldots, S_p\}$. It is assumed that the small sensors can (collectively) search only a subset of the AOI for the object (Figure B.1). In general, the measurements collected from S are allowed to follow any exponential family distribution while measurements from a small sensor S_i will be required to follow a Bernoulli distribution. The idea of sensor output following a Bernoulli distribution can be a natural assumption. Some sensors produce measurements with a natural Bernoulli interpretation. EGO sensors, for example, measure the air-to-fuel ratio in the exhaust gas and determine when this ratio crosses a threshold. As another example, Kim et al. (2005) consider binary proximity sensors with varying ranges. Other times, sensor output is quantized and transmitted to a data fusion center in the form of binary data (Shen et al. 2014).

To combine the information obtained from all sensors we propose a ML-based approach. In particular, it is assumed that n_i i.i.d. measurements are collected from S_i and that n i.i.d. measurements are collected from S. Estimating the parameters governing measurement distributions will give us an indication of whether an object is present or not; details are given in Section B.2. Based on the theory by Spall (2014), Maranzano and Spall (2011), and Spall (2009), convergence and asymptotic normality results are readily available. It is important to mention that the theory does not require sample sizes to be the same for all sensors (nor to increase at the same rate).

APPENDIX B. SYSTEM IDENTIFICATION FOR SENSOR DATA FUSION

In practice, the existence of small and large sensors may be a reasonable assumption to make. For example, there exist sensors with different ranges (e.g., Kim et al. 2005) which allows for a natural interpretation of small vs. large sensors. As another (more abstract) example, Duffield (2005) considers a network tomography problem where the objective is to detect faulty links between nodes. Sampling from the large sensor here could correspond to collecting measurements of global network performance (i.e., whether the network can be classified as faulty or not). For example, it might be possible to detect whether there is an unusual mount of messages being transmitted between two (not necessarily adjacent) nodes. A faulty system might then correspond to one where too many messages are reaching a given node. The state of the full system could then be modeled using a Bernoulli random variable. Furthermore, each link in the network could also be tested to determine whether it is functioning properly; tests on individual links would correspond to small sensor measurements. Similar to the fault detection problem in an TTS, several applications to system reliability could fit into this small– and large sensor setting (e.g., Spall 2009).

In the remainder of this chapter: Section B.1 introduces the general detection problem and the two motivating applications, Section B.2 discusses how how sensor information is combined and used for detection, Section B.3 contains numerical results, and Section B.4 contains some final comments.

B.1 Preliminaries

Two main detection examples will be treated. The first is a search problem where the objective is to test for the presence and location of a static object. The second problem is a fault detection problem for an TTS. It is important to note that the main detection problem is more general than either of these examples and is described next.

B.1.1 The General Detection Problem

In general AOI can be thought of as a grid with p disjoint cells $\{C_1, \ldots, C_p\}$. Cell C_i may be inspected by sensor S_i while the large sensor S may inspect the entire AOI. Additionally, one of the assumptions made is that measurements from S_i follow a Bernoulli distribution. Letting ρ_i be the parameter governing said distribution we define the vector $\theta \equiv [\rho_1, \ldots, \rho_p]^\top$. Similarly, if Y is a measurement from sensor S then $\rho \equiv E[Y]$. It is assumed that n_i i.i.d. samples are collected from S_i and n i.i.d. samples are collected from S. Furthermore, samples ontained from different sensors are independent of each other. We let X_i denote the sum of the n_i measurements from S_i and Y_k denote the kth measurement from S (for $k = 1, \ldots, n$). The goal is to detect the presence (or absence) of a static object within the AOI. Specifically, we would like to know whether an object is present in any of the given cells. The two detection examples are described next.

B.1.2 The Search Problem

Here the objective is to determine whether certain static object is present within a given search area. In this scenario the AOI is a physical space (for simplicity we let it be two-dimensional). It should be noted that the grid representation of the AOI is merely conceptual and the location of the different sensors may resemble Figure B.1. It is, however, assumed that cells do not intersect; a reasonable assumption when the small sensors are placed far enough apart. Setting sensors so that they are situated away from each other leads to situations where some cell is never inspected; this is permitted by the asymptotic theory in Spall (2014). Without loss of generality we can then assume:

$$\bigcup_{i=1}^{p} C_i = \text{AOI}.$$

The case where the union of all cells is not equal to the AOI will not be considered. Observe that if an object is located within the AOI, then it is located in at least one cell and vice versa. Here the distribution of Y (a measurement from the large sensor S) is also assumed to have a Bernoulli distribution.

The second example to consider is the detection of leaks in an TTS. The TTS considered by Zhou et al. (2012) is as follows: three fluid-filled tanks ($T_1, T_2,$ and T_3) are connected in series via a set of pipes. Water flows from T_1 to T_2 to T_3 and finally exits into a reservoir. All tanks are T_1 and T_3 are equipped

with pressure sensors whose measurements are converted into liquid levels. Furthermore, two pumps, Q_1 and Q_3, are fitted into the same two tanks and provide the flow rate. All tanks are fitted with adjustable valves to simulate clogs and leaks. The problem is then that of leakage-fault diagnosis (the detection of leaky tanks in the system) based on liquid level observations. The task is designing a test for fault detection that is sensitive to the fault and yet tolerant to noise and errors in the system's dynamics.

B.1.3 The Leakage Fault Diagnosis Problem

Liquid levels alone are not sufficient for detecting leaky tanks; the dynamics governing the levels must be modeled. Specifically, let $h_k^{(i)}$ denote the liquid level of tank T_i at time k and define:

$$\boldsymbol{x}_k \equiv \begin{bmatrix} h_k^{(1)} \\ h_k^{(2)} \\ h_k^{(3)} \end{bmatrix}, \quad \boldsymbol{y}_k \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k$$

where $\boldsymbol{y}_k$ denotes the observable liquid levels (only T_1 and T_3 are equipped with pressure sensors). The discrete-time dynamics of the state vector $\boldsymbol{x}_k$ are modeled according to:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}_u\boldsymbol{u}_k + \boldsymbol{B}_d\boldsymbol{d}_k + \boldsymbol{B}_f\boldsymbol{f}_k \tag{B.1}$$

where $\boldsymbol{A}, \boldsymbol{B}_u, \boldsymbol{B}_d, \boldsymbol{B}_f$ are matrices, $\boldsymbol{u}_k$ is a set of controls (essentially this controls the flow rate of Q_1 and Q_3), $\boldsymbol{d}_k$ is a disturbance caused by liquid fluctuations and $\boldsymbol{f}_k$ is a possible leakage fault (if $\boldsymbol{f}_k = 0$ then there are no faults in the system). $\boldsymbol{x}_k, \boldsymbol{d}_k$ and $\boldsymbol{f}_k$ are three-dimensional vectors while $\boldsymbol{u}_k$ is two-dimensional (there are only two pumps). Theoretically, one could compare the observed state vector $\boldsymbol{x}_{k+1}$ to a predicted $\tilde{\boldsymbol{x}}_{k+1}$ generated using (B.1) with $\boldsymbol{f}_k = 0$. If the error between the predicted and observed state-vectors is too large it would give an indication that there exists a fault somewhere in the system. This is the general idea behind the work in Zhou et al. (2012). The authors give a precise definition if the error between predicted and observed states. This error is referred to as the *residual*. The authors also provide specific thresholds that determine when the residual is too large. Furthermore, it is assumed that at most one tank is faulty at any given time and a precise threshold-based methodology for detecting the faulty tank is given.

The relationship between the work by Zhou et al. (2012) and our search problem can now be illustrated. At a given time the residual is tested against a threshold. If the residual exceeds the threshold then the system is classified as faulty (this is analogous to a large sensor measurement). Furthermore, when attempting to locate the faulty tank a three-dimensional vector $\boldsymbol{v}$ of Bernoulli responses is produced. For example, if T_1 is classified as faulty then $\boldsymbol{v} = [v^{(1)}, v^{(2)}, v^{(3)}]^\top = [1, 0, 0]^\top$. Note, however, that the $v^{(i)}$ are not independent

of each other due to the assumption that at most one tank is faulty. Recall that in this paper we require samples from different small sensor to be independent. For this reason, we assume any single tank may be faulty independently of whether the other two tanks are faulty.

To illustrate our proposed methodology for combining sensor information we consider a simplified problem. We will assume that all tanks are equipped with pressure sensors and that the state-vector $\boldsymbol{x}_k$ evolves according to (B.1) with $\boldsymbol{d}_k = 0$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}_u\boldsymbol{u}_k + \boldsymbol{B}_f\boldsymbol{f}_k. \tag{B.2}$$

In addition, we let $\boldsymbol{z}_k$ be an observable random variable such that:

$$\boldsymbol{z}_k = \begin{bmatrix} z_k^{(1)} \\ z_k^{(2)} \\ z_k^{(3)} \end{bmatrix} \equiv \boldsymbol{x}_k + \boldsymbol{\varepsilon}_k$$

where $\boldsymbol{\varepsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. Lastly, $\boldsymbol{x}_k$ is not observable.

If we assume that $\boldsymbol{f}_k = 0$ and that $\boldsymbol{u}_k$ is known and constant during a given time period (constant for $k \in T \equiv \{t, \ldots, t+\tau\}$ for some $t, \tau \geq 1$) it is easy to obtain confidence intervals for $z_{k+1}^{(i)}$ and for $\ell_{k+1} \equiv \sum_{i=1}^{3} z_{k+1}^{(i)}$. If ℓ_{k+1} lies outside its confidence interval we conclude $\boldsymbol{f}_k \neq 0$ and a fault is detected. Furthermore,

if $h_{k+1}^{(i)}$ lies outside its confidence interval then a fault is detected in tank T_i. Note that for all $k \in T$ the probability of setting of the fault detection alarm is constant whenever $\boldsymbol{f}_k = \mathbf{0}$. However, the same cannot be said when $\boldsymbol{f}_k \neq \mathbf{0}$. For this reason, we cannot assume that ρ_i is independent of k. Motivated by the fact that the final goal is that of classification and not estimation, we choose to ignore this fact; our numerical experiments are performed assuming ρ_i is constant for $k \in T$.

In the notation of the general detection problem (Section B.1.1) we define Y_k as the indicator of whether ℓ_k is outside its confidence interval and let X_i be the number of times a fault was detected in T_i for $k \in T$. Therefore, $n = n_i = |T|$.

B.2 Methodology

The ML based approach for detection consists of estimating the true parameters θ^* and ρ^* and then using these estimates for detection. Establishing a relationship between estimation and object detection relies on an important assumption: $\rho_i > 0.5$ whenever the object is present in C_i and $\rho_i \leq 0.5$ otherwise. This bound is an arbitrary but intuitive choice. The bound implies that if a sensor appears to detect an object more often then not then it must be that an object is actually present. Without any other information about S_i this is the least we can hope for. If Y also follows a Bernoulli distribution (e.g., the search

problem from Section B.1.2 and the detection problem from Section B.1.3) then the same logic applies to ρ. In the two examples to be considered here, the false positive (FP) and true positive (TP) rates are assumed known for all sensors. In this, case an object is said to be detected in C_i if the estimated value of ρ_i is closer to the TP than to the FP rate of S_i.

To further illustrate the basic idea behind our classification task, consider two coins, the first coin has a known probability of heads equal to 0.8 while the second has a known probability of heads equal to 0.3. If one is to choose a single coin with equal probability then the unconditional probability of heads after a single flip is 0.55. However, when collecting data from sample tosses it is implicit we must first choose a coin. As sample sizes increase the MLE of the data will converge to either 0.8 or 0.3. Therefore, once the data has been collected we can use this information to infer which coin has been chosen. Here, choosing the first coin may be analogous to having an object be present while choosing the second is the complementary situation.

Given the collection of measurements obtained from the different sensors, estimating θ and ρ could be done using two fundamentally different approaches:

- **Independent Estimation:** Here ρ as well as each of the parameters $\{\rho_1, \ldots, \rho_p\}$ are estimated independently.
- **Joint Estimation:** A relationship between ρ and θ is exploited so that joint parameter estimation can be used.

Here we will focus on the case of joint parameter estimation. Joint estimation will allow us to obtain detection decisions that are consistent between the small and large sensors. As an example, consider the search problem (Section B.1.2) where $Y \sim$ Bernoulli(ρ) and with $n_i = n = 1$ for all i. Furthermore, assume all of the small sensors failed to detect an object while the large sensor succeeded in detecting it (i.e., $X_i = 0$ for all i and $Y_1 = 1$). The ML estimates when using independent estimation would be $\hat{\theta}_{\text{ind}} = [0, \ldots, 0]^\top$ and $\hat{\rho}_{\text{ind}} = 1$ (the subindex "ind" is used to indicate that independent estimation was used). This might lead us to conclude that an object is present within the AOI but not located within any of the cells which is a contradiction. Using joint estimation implies we wish to find and exploit a relationship between ρ and θ. In particular, we will assume there exists a function h such that $\rho = h(\theta)$. We will derive this function for two special cases. Next we give the details behind the joint estimation approach.

Since the distribution $p(Y = y|\rho)$ of Y belongs to the exponential family, we have $p(Y = y|\rho) = \exp[a(\rho) + b(\rho) + c(y)]$ for some functions $a(\rho)$, $b(\rho)$ and $c(y)$. In addition, the log-likelihood function with respect to θ is given by:

$$\mathcal{L}(\theta) = \sum_{k=1}^{n} [a(\rho) Y_k + b(\rho)] + \sum_{i=1}^{p} [X_i \log(\rho_i) + (n_i - X_i) \log(1 - \rho_i)] + \text{constant},$$

where the constant term does not depend on θ. In some situations the MLE for

θ can be obtained by finding the roots of the score function (the score function is the gradient of the log-likelihood function):

$$\frac{\mathcal{L}(\theta)}{\partial\theta} = \left[\sum_{k=1}^{n} a'(\rho)(Y_k - \rho)\right] \boldsymbol{h}'(\theta) + \begin{bmatrix} \frac{X_1}{\rho_1} - \frac{n_1 - X_1}{1-\rho_1} \\ \vdots \\ \frac{X_p}{\rho_p} - \frac{n_p - X_p}{1-\rho_p} \end{bmatrix}. \tag{B.3}$$

However, it is not generally true that the root of the score equation coincides with the MLE or that the root necessarily exists nor is unique. Still, it is common practice to solve the score equation to attempt to obtain the MLE. This is the approach taken here. Next we briefly review some of the existing theory regarding a solution to (B.3).

Define $N \equiv n + n_1 + \cdots + n_p$ and let $\hat{\theta}^{(N)}$ be the value of θ that solves (B.3) assuming it equals the MLE; the value of $\hat{\theta}^{(N)}$ can be obtained by using the relationship $\rho = h(\theta)$. Let θ^* and ρ^* be the true parameters to be estimated. Furthermore, define $\hat{\rho}^{(N)} \equiv h(\hat{\theta}^{(N)})$. Spall (2014) shows that under some regularity conditions we have the following results: Spall (2014 Theorem 3.1) defines n_s as the slowest increasing sample size among the small sensors and shows that if $n + n_s \to \infty$ then $\hat{\rho}^{(N)} \to \rho^*$ w.p.1. A similar result (Spall 2014, Theorem 3.2) establishes the convergence w.p.1 of $\hat{\theta}^{(N)}$ to θ^* whenever $n_i \to \infty$ for at least $p-1$ of the small sensors. Asymptotic normality results are also presented in (Spall 2014, Theorems 4.1 and 4.2). Specifically, $\sqrt{n^+}(\hat{\rho}^{(N)} - \rho^*) \xrightarrow{\text{dist}} N\left(0, \sigma(n_{.s}^*)^2\right)$ as

$N \to \infty$ where $n^+ \equiv n + n_s$; we refer the reader to (Spall 2014) for the definition of $\sigma(\cdot)$ and $n_{.s^*}$. Similarly:

$$\boldsymbol{S}_N(\hat{\theta}^{(N)} - \theta^*) \xrightarrow{\text{dist}} N(\mathbf{0}, \boldsymbol{\Sigma}_{\theta^*})$$

as $N \to \infty$ where $\boldsymbol{S}_N$ is a matrix depending on the n_i and $\boldsymbol{\Sigma}_{\theta^*}$ is a limiting function of $\boldsymbol{S}_N$ and of the Fisher information matrix at θ^*. The reason we include these results here is to highlight the dependence on the sample sizes (which may vary among sensors) and the rate at which they increase.

An important part in implementing ML estimation via (B.3) is the identification of the function h relating θ to ρ. This function is problem dependent. Next we show how to obtain h for the search- and fault detection problems introduced in Sections (Section B.1.2) and (Section B.1.3), respectively.

B.2.1 Determining h for the search problem (Section B.1.2)

Here, the large sensor S has a Bernoulli(ρ) distribution where $0 < \rho < 1$. Let $\rho_i^{\text{FP}}, \rho_i^{\text{TP}}, \rho^{\text{FP}}$ and ρ^{TP} denote the FP and TP rates for sensors S_i and S. We have the following relationships:

$$\rho_i = \pi_i \rho_i^{\text{TP}} + (1 - \pi_i)\rho_i^{\text{FP}}, \tag{B.4a}$$

$$\rho = \pi \rho^{\text{TP}} + (1 - \pi)\rho^{\text{FP}}. \tag{B.4b}$$

$\{\rho_i^{\mathrm{FP}}, \rho_i^{\mathrm{TP}}\}$	$\equiv$	False positive & true positive rates of S_i.
π_i	$\equiv$	Indicator of whether an object is present in C_i.
$\{\rho^{\mathrm{FP}}, \rho^{\mathrm{TP}}\}$	$\equiv$	False positive & true positive rates of S.
π	$\equiv$	Indicator of whether an object is present in the AOI.
$\boldsymbol{\theta}$	$\equiv$	$[\rho_1, \ldots, \rho_p]^\top$.
ρ_i	$\equiv$	Mean of the distribution of measurements from S_i.
ρ	$\equiv$	Mean of the distribution of measurements from S.
$\boldsymbol{\theta}^*$	$\equiv$	True small sensor parameter vector.
ρ^*	$\equiv$	True large sensor parameter.

Table B.1: A list of useful notation.

where π_i is the indicator function of whether an object is located in C_i and π is the indicator function of whether and object is located somewhere within the AOI (see Table B.1). The TP and FP rates are assumed known, an assumption often satisfied when there exists a probabilistic model (such as for the aforementioned acoustic proximity sensors). Alternatively, these rates may also be obtained via experimentation. Niu et al. (2005) address the issue of estimating false positive detection rates for a particular data fusion technique.

Via a simple manipulation of (B.4a) we have:

$$\pi_i = \frac{\rho_i - \rho_i^{\mathrm{FP}}}{\rho_i^{\mathrm{TP}} - \rho_i^{\mathrm{FP}}}; \tag{B.5}$$

we will also assume that $\rho_i^{\mathrm{TP}} \neq \rho_i^{\mathrm{FP}}$ and that $\rho^{\mathrm{TP}} \neq \rho^{\mathrm{FP}}$, otherwise sensors would not be providing the information we require for our final classification task.

Next, the indicator function of an object being in the entire search area is equal to $\pi = 1 - \prod_{i=1}^{p}(1 - \pi_i)$ so that using (B.5) we obtain the following

relationship between π, ρ_i^{TP}, and ρ_i^{FP}:

$$\pi = 1 - \prod_{i=1}^{p} \left(1 - \frac{\rho_i - \rho_i^{\text{FP}}}{\rho_i^{\text{TP}} - \rho_i^{\text{FP}}} \right).$$

Consequently, combining the previous result with (B.4b) gives the desired h:

$$\rho = \underbrace{\rho^{\text{TP}} + (\rho^{\text{FP}} - \rho^{\text{TP}}) \prod_{i=1}^{p} \left(1 - \frac{\rho_i - \rho_i^{\text{FP}}}{\rho_i^{\text{TP}} - \rho_i^{\text{FP}}} \right)}_{\equiv h(\theta)}. \tag{B.6}$$

B.2.2 Determining h for the fault detection problem (Section B.1.3)

Finding a function to relate ρ to θ is complicated. However, if the TP rates are small, (B.6) yields the approximation:

$$\rho \approx \underbrace{\rho^{\text{FP}} \prod_{i=1}^{p} \left(1 + \frac{\rho_i - \rho_i^{\text{FP}}}{\rho_i^{\text{FP}}} \right)}_{\equiv h(\theta)}. \tag{B.7}$$

Therefore, (B.7) is the approximation to be used in the numerical experiments. It is important to notice that the true positive rate may not be small enough for the approximation to hold. This is something we investigate numerically.

B.3 Numerical Analysis

For the search problem two disjoint cells constituted the AOI. ($p = 2$). Each of the cells corresponds to one of the C_is. An object was assumed to be located

within cell C_1. Furthermore, the FP and FN rates were assumed equal for all of the small sensors. Sample sizes of $n = n_i = 5, 10, 15, 20$ were used and four estimates were produced: $\hat{\theta}^{(N)}$, $\hat{\rho}^{(N)}$, $\hat{\theta}_{\text{ind}}$, and $\hat{\rho}_{\text{ind}}$. The first two variables are estimates for θ^* and ρ^* obtained using joint estimation while the last two variables are estimates obtained via independent estimation. To compare performance in estimating ρ^* a relative error was employed:

$$e_\rho^{(1)} \equiv \frac{|\hat{\rho}^{(N)} - \rho^*|}{\rho^*},\ e_\rho^{(2)} \equiv \frac{|\hat{\rho}_{\text{ind}} - \rho^*|}{\rho^*}.$$

Similarly, the error in estimating θ^* was measured using the errors:

$$e_\theta^{(1)} \equiv \frac{\|\hat{\theta}^{(N)} - \theta^*\|}{\|\theta^*\|},\ e_\theta^{(2)} \equiv \frac{\|\hat{\theta}_{\text{ind}} - \theta^*\|}{\|\theta^*\|}.$$

Table B.2 compares the mean (taken over 100 replications) and standard deviation (SD) of $e_\rho^{(1)}$ and $e_\rho^{(2)}$ for $\rho^* = 0.8, \rho_1^* = 0.6$ and $\rho_i^* = 0.4$ for $i \neq 1$. Furthermore, the FP and TP rates used were $\rho^{\text{FP}} = 0.2$, $\rho_i^{\text{FP}} = 0.4$, $\rho^{\text{TP}} = 0.8$ and $\rho_i^{\text{TP}} = 0.6$. It is shown that joint parameter estimation outperforms independent estimation. Table B.2 also compares the mean and SD of $e_\theta^{(1)}$ and $e_\theta^{(2)}$ for the same parameter settings. Once again it appears joint estimation outperforms independent estimation. Still, more extensive numerical experiments are required.

Because estimating the true parameters is important in determining if the object is present or not it appears that joint parameter estimation would im-

$n = n_i$	Mean of $e_\rho^{(1)}$	Mean of $e_\rho^{(2)}$	SD of $e_\rho^{(1)}$	SD of $e_\rho^{(2)}$
5	0.09	0.18	0.13	0.15
10	0.06	0.12	0.08	0.10
15	0.05	0.09	0.06	0.08
20	0.05	0.09	0.06	0.08
$n = n_i$	Mean of $e_\theta^{(1)}$	Mean of $e_\theta^{(2)}$	SD of $e_\theta^{(1)}$	SD of $e_\theta^{(2)}$
5	0.16	0.30	0.13	0.18
10	0.12	0.24	0.11	0.13
15	0.11	0.20	0.12	0.11
20	0.10	0.18	0.10	0.10

Table B.2: Mean and SD of $e_\rho^{(1)}$, $e_\rho^{(2)}$, $e_\theta^{(1)}$, and $e_\theta^{(2)}$.

prove the classification task. However, it turns out that the number of misclassified cells was (on average) comparable for both methods (i.e., joint- and independent estimation) whenever the small sensors were "good enough". For the TTS leak detection problem the classification and estimation errors were comparable for both methods.

B.4 Concluding Remarks

We have presented a general idea of how to combine information from small and large sensors to aid detection. For the search area problem the numerical experiments indicate that estimation of the parameters governing the sensors was improved using our methodology. Still, the task of determining whether an object was present or not in the AOI presented a similar behavior whenever the small sensors were informative enough (i.e., when the small sensors

had high TP and low FP rates). However, one important thing to take into consideration is that jointly estimating the measurement parameters, as in our proposed methodology, could help infer whether an object is present in regions that are never inspected by any small sensor. For the fault-detection problem the numerical experiments did not indicate improved estimation- or classification performance. The most likely explanation is that there is only a significant advantage when sample sizes are very small and the error rates for many sensors are large. However, for small sample sizes the score function may have multiple solutions. Further analysis could include theoretical analysis regarding whether combining information can significantly improve classification, especially when the large sensor measurements are not binary.

Frequently Used Notation

A list of the most commonly used notation arranged by category. The notation in this list is consistent throughout the entire dissertation (with the exception of Section 1.2 where some of the notation was borrowed from the references being surveyed).

General Notation

$\equiv$ is the symbol for "defined as".

$A \cup B$ denotes the union of the sets A and B.

$A \cap B$ denotes the intersection of the sets A and B.

S^c denotes the complement of the set S.

$\emptyset$ denotes the empty set.

$\square$ denotes the end of a proof.

$\binom{a}{b}$ represents the quantity $a!/[(a-b)!b!]$.

$\mathbb{Z}^+$ denotes the set of strictly positive integers.

$\mathbb{R}^p$ denotes the Euclidean space of dimension p.

w.l.o.g. means "without loss of generality".

$\lceil a \rceil$ is the ceiling function applied to a real number a.

$\lfloor a \rfloor$ is the floor function applied to a real number a.

I_T denotes the interval $[-T, T]$ for $T > 0$.

$O(\cdot)$ denotes the standard big-*O* notation.

$o(\cdot)$ denotes the standard little-*o* notation.

Matrices and Vectors

Bolded variables represent either vectors or matrices.

$\boldsymbol{A}^\top$ denotes the transpose of the matrix $\boldsymbol{A}$.

$\mathrm{trace}\,(\boldsymbol{A})$ is the trace of the matrix $\boldsymbol{A}$.

$\|\cdot\|$ represents the Frobenius norm.

$\boldsymbol{I}$ denotes the identity matrix of unspecified dimensions.

M_{ij} (also $(\boldsymbol{M})_{ij}$ or $[\boldsymbol{M}]_{ij}$) denotes the (i,j)th entry of the matrix $\boldsymbol{M}$.

$M_{k(ij)}$ denotes the (i,j)th entry of the matrix $\boldsymbol{M}_k$.

v_j (also $(\boldsymbol{v})_j$ or $[\boldsymbol{v}]_j$) denotes the jth entry of the vector $\boldsymbol{v}$.

$v_{k(j)}$ denotes the jth entry of the vector $\boldsymbol{v}_k$.

p is reserved for the dimension of θ so that $\theta = [\tau_1, \ldots, \tau_p]^\top \in \mathbb{R}^p$.

Probability

$\chi\{\mathcal{E}\}$ denotes the indicator function of the event $\mathcal{E}$.

$P(\mathcal{E})$ denotes the probability of the event $\mathcal{E}$.

$E[\mathcal{X}]$ represents the expectation of the random variable $\mathcal{X}$.

$\mathrm{Var}\,(\mathcal{X})$ is the variance or covariance matrix of $\mathcal{X}$.

$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a normal random variable with mean $\boldsymbol{\mu}$ and variance/covariance $\boldsymbol{\Sigma}$.

Bernoulli(p) denotes a Bernoulli random variable with parameter p.

i.i.d. means "independent and identically distributed".

w.p.1 means "with probability one".

$\xrightarrow{\text{dist}}$ means convergence in distribution.

$\mathcal{X} \sim$ is used to indicate that the random variable $\mathcal{X}$ has certain distribution.

$(\Omega, \mathcal{F}, P)$ is a probability space with sample space Ω, σ-field $\mathcal{F}$, and measure P.

Sequences and Sums

$g_n = o(h_n)$ if $h_n^{-1} g_n \to 0$ as $n \to \infty$.

$G_n = O(h_n)$ if $\|h_n^{-1} G_n\| \leq c$ for some $c \in \mathbb{R}$ and all n.

$\sum_{i=a}^{b} (\cdot)_i = 0$ whenever $b < a$.

Stochastic Optimization

$L(\theta)$ denotes a loss function to minimize.

θ denotes the vector of parameters being estimated.

$\boldsymbol{g}(\theta)$ denotes the gradient vector of $L(\theta)$.

$\boldsymbol{H}(\theta)$ denotes the Hessian matrix of $L(\theta)$.

θ^* denotes a minimizer of $L(\theta)$ or a solution to $\boldsymbol{g}(\theta) = \mathbf{0}$.

$\hat{\theta}_k$ is the iterate produced in the kth iteration of an algorithm searching for θ^*.

$\hat{\boldsymbol{g}}_k(\theta)$ denotes a noisy gradient measurement obtained during iteration $k+1$.

$Q(\theta, \boldsymbol{V})$ denotes a noisy measurement of $L(\theta)$.

GCSA-Specific Notation

$\mathcal{S}_j$ for $j = 1, \ldots, d$ is a set of coordinates defining a subvector of θ (see p. 44).

$j_k(m)$ is a random variable in $\{1, \ldots, d\}$.

$\boldsymbol{v}^{(j)}$ for $\boldsymbol{v} \in \mathbb{R}^p$ forces certain entries of $\boldsymbol{v}$ to be zero according to (3.20).

$\hat{\theta}_{\boldsymbol{k}}^{(I_{m,i})}$ denotes an intermediate step within an iteration (see p. 42).

$\boldsymbol{J}^{(j)}(\theta)$ denotes the Jacobian of $\boldsymbol{g}^{(j)}(\theta)$.

$a_k^{(j)}$ for $j = 1, \ldots, d$ denotes a gain sequence.

s_k is the random number of blocks in the $(k+1)$st iteration.

$n_k(m)$ is the number of updates in the mth block of the $(k+1)$st iteration.

$\tilde{A}_k(m) = \sum_{j=1}^{d} \mathcal{X}\{j_k(m) = j\}\tilde{a}_k^{(j)}$ (see p. 45).

$\tilde{a}_{k+1}^{(j)} = a_0^{(j)} + \sum_{i=0}^{k} \mathcal{X}\{\varphi_k^{(j)} \geq i\}(a_{i+1}^{(j)} - a_i^{(j)})$ (see p. 47).

$\varphi_k^{(j)} = \sum_{i=0}^{k} \mathcal{X}\left\{(\sum_{m=1}^{s_i} \mathcal{X}\{j_i(m) = j\}) > 0\right\} - 1$ (see p. 46).

$x_k(j) = (\tilde{a}_k^{(j)}/a_k) \sum_{m=1}^{s_k} \mathcal{X}\{j_k(m) = j\} n_k(m)$ (see p. 52).

$\mu_k(j) = E[x_k(j)]$ (see p. 52).

$\mu(j) = \lim_{k\to\infty} \mu_k(j)$ (see p. 52).

$X_k = \sum_{j=1}^{d} x_k(j)$ (see p. 59).

$\boldsymbol{h}_k(\theta) = \sum_{j=1}^{d} \mu_k(j)\boldsymbol{g}^{(j)}(\theta)$ (see p. 53).

$\boldsymbol{h}(\theta) = \sum_{j=1}^{d} \mu(j)\boldsymbol{g}^{(j)}(\theta)$ (see p. 53).

Bibliography

Abounadi, J., Bertsekas, D. P., & Borkar, V. (2002). Stochastic approximation for nonexpansive maps: application to Q-learning algorithms. *SIAM Journal on Control and Optimization*, *41*(1), pp. 1–22.

Absil, P. A., & Kurdyka, K. (2006). On the stable equilibrium points of gradient systems. *Systems & Control Letters*, *55*(7), pp. 573–577.

Andrews, M. (2006). Joint optimization of scheduling and congestion control in communication networks. In *Proceedings of the Conference on Information Sciences and Systems (CISS)*, (pp. 1572–1577).

Benveniste, A., Métivier, M., & Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag.

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical methods*. Prentice-Hall.

BIBLIOGRAPHY

Bhatnagar, S. (2011). The Borkar–Meyn theorem for asynchronous stochastic approximations. *Systems & Control Letters*, *60*(7), pp. 472–478.

Bhatnagar, S., Prasad, H. L., & Prashanth, L. A. (2013). *Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods*. Springer.

Bianchi, P., & Jakubowicz, J. (2013). Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, *58*(2), pp. 391–405.

Bickel, P. J., & Doksum, K. A. (2007). *Mathematical Statistics*. New Jersey: Pearson, 2nd ed.

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

Borkar, V. S. (1997). Stochastic approximation with two time scales. *Systems & Control Letters*, *29*(5), pp. 291–194.

Borkar, V. S. (1998). Asynchronous stochastic approximations. *SIAM Journal on Control and Optimization*, *36*(3), pp. 840–851.

Borkar, V. S., & Meyn, S. P. (2000). The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, *38*(2), pp. 447–469.

BIBLIOGRAPHY

Botts, C. H., Spall, J. C., & Newman, A. J. (2016). Multi-agent surveillance and tracking using cyclic stochastic gradient. In *Proceedings of the American Control Conference (ACC)*, (pp. 270–275).

Box, G. E. P., & Wilson, K. B. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society*, *13 (Series B)*, pp. 1–38.

Canutescu, A. A., & Dunbrack, R. L. (2003). Cyclic coordinate descent: a robotics algorithm for protein loop closure. *Protein Science*, *12*(5), pp. 963–972.

Chung, K. L. (2001). *A Course in Probability Theory*. Elsevier, 3rd ed.

Chung, T. H., & Burdick, J. W. (2012). Analysis of search decision making using probabilistic search strategies. *IEEE Transactions on Robotics*, *28*(1), pp. 132–144.

Cronin, J. (2007). *Ordinary Differential Equations: Introduction and Qualitative Theory*. Florida: CRC Press, 3rd ed.

Dennis, J. E., & Schnabel, R. B. (1989). A view of unconstrained optimization. *Handbooks in Operations Research and Management Science*, *1*, pp. 1–72.

Doob, J. L. (1953). *Numerical Optimization*. New York: John Wiley & Sons, 2nd ed.

BIBLIOGRAPHY

Duffield, N. (2006). Network tomography of binary network performance characteristics. *IEEE Transactions on Information Theory*, *52*(12), pp. 5373–5388.

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

Fabian, V. (1968). On asymptotic normality in stochastic approximation. *Annals of Mathematical Statistics*, *39*(4), pp. 1327–1332.

Herceg, M., Kvasnica, M., Jones, C., & Morari, M. (2013). Multi-parametric toolbox 3.0. In *Proceedings of the European Control Conference (ECC)*, (pp. 502–510). Zürich, Switzerland. `http://control.ee.ethz.ch/~mpt`.

Hernandez, K. (2015). Combined sensor information for detection. In *Proceedings of the Conference on Information Sciences and Systems (CISS)*, (pp. 1–6).

Hernandez, K. (2016). Cyclic stochastic optimization via arbitrary selection procedures for updating parameters. In *Proceedings of the Conference on Information Sciences and Systems (CISS)*, (pp. 349–354).

Hernandez, K., & Spall, J. C. (2014). Cyclic stochastic optimization with noisy function measurements. In *Proceedings of the American Control Conference (ACC)*, (pp. 5204–5209).

BIBLIOGRAPHY

Hernandez, K., & Spall, J. C. (2016). Asymptotic normality and efficiency analysis of the cyclic seesaw stochastic optimization algorithm. In *Proceedings of the American Control Conference (ACC)*, (pp. 7255–7260).

Hernandez, K., & Spall, J. C. (2017). A complete result on the asymptotic normality of stochastic approximation algorithms. *(Under review)*.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, *58*(301), pp. 13–30.

Horn, R. A., & Johnson, C. R. (2010). *Matrix Analysis*. New York: Cambridge University Press.

Hu, J., Hu, P., & Chang, H. S. (2012). A stochastic approximation framework for a class of randomized optimization algorithms. *IEEE Transactions on Automatic Control*, *57*(1), pp. 165–178.

Kar, S., Moura, J. M., & Poor, H. V. (2013). Distributed linear parameter estimation: asymptotically efficient adaptive strategies. *SIAM Journal on Control and Optimization*, *51*(3), pp. 2200–2229.

Kim, W., Mechitov, K., Choi, J. Y., & Ham, S. (2005). On target tracking with binary proximity sensors. In *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*, (pp. 301–308).

BIBLIOGRAPHY

Knight, N., Carson, E., & Demmel, J. (2013). Exploiting data sparsity in parallel matrix powers computations. In *Proceedings of the International Conference on Parallel Processing and Applied Mathematics*, (pp. 15–25).

Konda, V. R., & Tsitsiklis, J. N. (1997). Convergence rate of linear two-time-scale stochastic approximation. *Annals of Applied Probability*, *14*(2), pp. 796–819.

Kushner, H., & Yin, G. (1997). *Stochastic Approximation Algorithms and Applications*. New York: Springer-Verlag.

Kushner, H. J., & Clark, D. S. (1978). *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York: Springer-Verlag, 2nd ed.

Lee, S., Diaz-Mercado, Y., & Egerstedt, M. (2015). Multirobot control using time-varying density functions. *IEEE Transactions on Robotics*, *31*(2), pp. 489–493.

Lee, S., & Park, F. C. (2008). Cyclic optimization algorithms for simultaneous structure and motion recovery in computer vision. *Engineering Optimization*, *40*(5), pp. 403–419.

Li, B., & Petropuli, A. (2014). Efficient target estimation in distributed MIMO

radar via the ADMM. In *Proceedings of the Conference on Information Sciences and Systems (CISS)*, (pp. 1–5).

Li, Y., & Osher, S. (2009). Coordinate descent optimization for ℓ_1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging*, *3*(3), pp. 487–503.

Ljung, L. (1977). Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, *22*(4), pp. 551–575.

Luo, Z. Q., & Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, *72*(1), pp. 7–35.

Luo, Z. Q., & Tseng, P. (1993). Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, *46*(1), pp. 157–178.

Maranzano, C. J., & Spall, J. C. (2011). Framework for estimating system reliability from full system and subsystem tests with dependence on dynamic inputs. In *Proceedings of the Conference on Decision and Control (CDC)*, (pp. 6666–6671).

Necoara, I. (2013). Random coordinate descent algorithms for multi-agent con-

vex optimization over networks. *IEEE Transactions on Automatic Control*, *58*(8), pp. 2001–2012.

Necoara, I., & Petrascu, A. (2014). A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, *57*(2), pp. 307–337.

Nedić, A., & Bertsekas, D. P. (2010). The effect of deterministic noise in subgradient methods. *Mathematical Programming*, *125*(1), pp. 75–99.

Nevel'son, M. B., & Has'minskii, R. Z. (1973). *Stochastic Approximation and Recursive Estimation*. Rhode Island: American Mathematical Society.

Niu, R., Varshney, P. K., & Cheng, Q. (2005). Distributed detection in a large wireless sensor network. *Information Fusion*, *7*(4), pp. 380–394.

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization*. New York: Springer, 2nd ed.

Øksendal, B. (2003). *Stochastic Differential Equations: an Introduction with Applications*. Vancouver: Springer, sixth ed.

Peterson, C. K., Newman, A. J., & Spall, J. C. (2014). Simulation-based examination of the limits of performance for decentralized multi-agent surveillance

and tracking of undersea targets. In *Proceedings of the SPIE Defense + Security Conference*, (pp. 90910–90915).

Ram, S. S., Nedić, A., & Veeravalli, V. V. (2009a). Incremental stochastic subgradient algorithms for convex optimization. *SIAM Journal on Optimization*, *20*(2), pp. 691–717.

Ram, S. S., Nedić, A., & Veervalli, V. V. (2009b). Asynchronous gossip algorithms for stochastic optimization. In *Proceedings of the Conference on Decision and Control held jointly with the 28th Chinese Control Conference (CDC/CCC)*, (pp. 3581–3586).

Ram, S. S., Nedić, A., & Veervalli, V. V. (2010). Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, *147*(3), pp. 516–545.

Renotte, C., & Wouwer, A. V. (2003). Stochastic approximation techniques applied to parameter estimation in a biological model. In *Proceedings of the International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, (pp. 261–265).

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

Rudin, W. (1976). *Principles of Mathematical Statistics*. McGraw-Hill, 3rd ed.

Shen, X., Varshney, P. K., & Zhu, Y. (2014). Robust distributed maximum likelihood estimation with dependent quantized data. *Automatica*, *50*(1), pp. 169–174.

Singh, C., Nedić, A., & Srikant, R. (2014). Random block-coordinate gradient projection algorithms. In *Proceedings of the Conference on Decision and Control (CDC)*, (pp. 185–190).

Solodov, M. V., & Zavriev, S. K. (1998). Error stability properties of generalized gradient-type algorithms. *Journal of Optimization Theory and Applications*, *98*(3), pp. 663–680.

Sommer, H. J. (Retrieved May 15 2017). polygeom.m. In *MATLAB Central File Exchange*. Last updated December 9 2016. `http://www.mathworks.com/matlabcentral/fileexchange/319-polygeom-m`.

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

Spall, J. C. (2003). *Introduction to Stochastic Search and Optimization*. New Jersey: John Wiley & Sons.

Spall, J. C. (2009). System reliability estimation and confidence regions from

subsystem and full system tests. In *Proceedings of the American Control Conference (ACC)*, (pp. 5067–5072).

Spall, J. C. (2012). Cyclic seesaw process for optimization and identification. *Journal of Optimization Theory and Applications*, *154*(1), pp. 187–208.

Spall, J. C. (2014). Identification for systems with binary subsystems. *IEEE Transactions on Automatic Control*, *59*(1), pp. 3–17.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, *109*(3), pp. 475–494.

Tseng, P., & Yun, S. (2009). Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, *140*(3), pp. 513–535.

Tsitsiklis, J. N. (1984). *Problems in Decentralized Decision making and Computation*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge Lab for Information and Decision Systems.

Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, *16*(3), pp. 185–202.

Tsitsiklis, J. N., Bertsekas, D. P., & Athans, M. (1986). Distributed asyn-

chronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, *31*(9), pp. 803–812.

Wang, I.-J., & Spall, J. C. (2008). Stochastic optimisation with inequality constraints using simultaneous perturbations and penalty functions. *International Journal of Control*, *81*(8), pp. 1232–1238.

Wang, Q., & Spall, J. C. (2011). Discrete simultaneous perturbation stochastic approximation on loss function with noisy measurements. In *Proceedings of the American Control Conference (ACC)*, (pp. 4520–4525).

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, *151*(1), pp. 3–34.

Xu, Y., & Yin, W. (2015). Block stochastic gradient iteration for convex and nonconvex optimization. *SIAM Journal on Optimization*, *25*(3), pp. 1686–1716.

Zhou, D., He, X., Wang, Z., Liu, G., & Ji, Y. (2012). Leakage fault diagnosis for an internet-based three-tank system: an experimental study. *IEEE Transactions on Control Systems Technology*, *20*(4), pp. 857–870.

Zhou, E., & Hu, J. (2014). Gradient-based adaptive stochastic search for non-differentiable optimization. *IEEE Transactions on Automatic Control*, *59*(7), pp. 1818–1832.

BIBLIOGRAPHY

Zorin, A., Mayer-Proschel, M., Noble, M., & Yakovlev, A. Y. (2000). Estimation problems associated with stochastic modeling of proliferation and differentiation of O-2A progenitor cells in vitro. *Mathematical Biosciences*, *167*(2), pp. 109–121.

Curriculum Vitae

Karla Hernández-Cuevas was born on November 1986 in Mexico City. Her interest in mathematics began in high-school after the local university, the University of Colima (Universidad de Colima), offered high-school seniors the opportunity to take mathematics and physics courses taught at an undergraduate-level. Her interest in applied mathematics deepened after interning at the Mathematical and Theoretical Biology Institute at Arizona State University where she worked on determining optimal treatment strategies for patients with mental disorders. After graduating form high-school she went on to obtain her Bachelor's degree in mathematics in 2009 from the University of Colima. Her undergraduate dissertation involved predicting the outcomes of UEFA (Union of European Football Associations) Champions League matches. After obtaining her Bachelor's degree Karla went on to intern at the Institute for Innovation and Technological Development (Instituto de Innovación y Desarrollo Tecnológico) at the University of Colima. There, she participated in a state-wide campaign to control the spread of mosquito-borne diseases.

CURRICULUM VITAE

In 2010 Karla was awarded a prestigious scholarship from the Mexican government's National Council of Science and Technology (CONACYT) to pursue her graduate studies at Johns Hopkins University (JHU). During her second year at JHU Karla was part of a research team providing consultation on statistics for the Howard Hughes Medical Institute's Janelia research center. In 2012 she obtained her Master's degree in Applied Mathematics & Statistics from JHU and began her doctoral research with Dr. James C. Spall also at JHU. Karla has given many talks and presentations at different conferences and universities on topics which include stochastic optimization, computer vision, optimal control, optimal design of experiments, and multi-sensor data fusion. She is the author of several peer-reviewed publications and has been a reviewer for several journals and conferences and for John Wiley & Sons.

During her graduate studies Karla took several courses on topics including statistics, probability, optimization, machine learning, computer vision, neural networks, applied optimal control, Monte Carlo methods, graph theory, bioinformatics, computational genomics, system identification, and stochastic optimization. She was also a teaching assistant for over a dozen courses, was the course instructor for two undergraduate courses in statistics, and served as the graduate representative of the Applied Mathematics & Statistics department before JHU's Graduate Representatives Organization.