Applications of Stochastic Gradient Descent for Optimization of Complex Systems with Uncertainty

by

Somak Das

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 7, 2021

Keywords: optimal control, stochastic optimization, thermal regulation, canonical decomposition, parallel factors, stochastic alternating least squares

Approved by

Hans Werner van Wyk, Assistant Professor of Department of Math and Stat
Yanzhao Cao, Professor and Associate Chair of Department of Math and Stat
Dmitry Glotov, Associate Professor and Associate Chair, Department of Math and Stat
Erkan Nane, Associate Professor, Department of Math and Stat
Luke Oeding, Associate Professor of Department of Math and Stat
Yang Zhou, Assistant Professor of Department of Computer Science and Software Engineering
Geroge Flowers, Dean of the Graduate School and Professor of Mechanical Engineering

Abstract

The Stochastic Gradient Descent (SGD) method in all its variations has gained popularity with the recent progress in Machine learning. In this dissertation we implement and analyze two modifications of the SGD method. In the first half of the dissertation we investigate the adaptive gradient descent (AdaGrad) method in the context to the optimal distributed control of parabolic partial differential equations with uncertain parameters. This stochastic optimization method achieves an improved convergence rate through adaptive scaling of the gradient stepsize. We prove the convergence of the algorithm for this infinite dimensional problem under suitable regularity, convexity, and finite variance conditions, and relate these to verifiable properties of the underlying system parameters. Finally, we apply our algorithm to the optimal thermal regulation of lithium battery systems under uncertain loads. In the second half of the dissertation we look at the design, implementation and analysis of Stochastic Alternating Least Squares (SALS) as a method that approximates the canonical decomposition of averages of sampled random tensors. Its simplicity and efficient memory usage make SALS an ideal tool for decomposing tensors in an online setting. We show, under mild regularization and readily verifiable assumptions on the boundedness of the data, that the SALS algorithm is globally convergent. Numerical experiments validate our theoretical findings and demonstrate the algorithm's performance and complexity.

Acknowledgments

I am extremely thankful to my co-advisors, Dr.Yanzhao Cao and Dr.Hans-Werner van Wyk for all the guidance, help and support they have provided during this journey. Without their wisdom and advice I would not have been able to complete my thesis. They have been very encouraging and inspiring every step of the way, reassuring that I had in me what it takes and providing me with insights essential to the design of this work. I am extremely grateful to Dr.Luke Oeding, Dr.Dmitry Glotov, Dr.Erkan Nane for being in my committee and Dr.Yang Zhou for being the University Reader. I thank all the colleagues and members of the Dept of Math and Stat who have created an excellent academic domain at Auburn University making it conducive to higher level education and research. I would also take this opportunity to thank my family and friends who have been very supportive and encouraging throughout this process.

Table of Contents

List of Figures

Chapter 1

Introduction

The idea of Artificial Intelligence (AI) began as a philosophical idea even before computers, when philosophers attempted to design human thinking using abstract mathematical logic [61]. Over the ages, the presence of AI can be found from ancient mythologies, to renaissance, to modern day science fictions that talk about humanoids indistinguishable from humans in intelligence and appearance [4, 50, 61]. With the advent of computers (machines capable of synthesizing mathematical logic), deep learning (process of computers learning complex concepts out of simpler concepts) can be traced back to the 1940s [23]. Scientists started serious consideration of building electronic brains that can imitate a living brain and human logic [16, 61].

In 1956, a group of scientists met at the Dartmouth Research Project on AI, which is known to have coined the term of artificial intelligence [42, 50]. The meeting conjectured the possibility of simulating the process of learning and other aspects of intelligence. Unfortunately in the 1970s AI faced extreme critiques and suffered setbacks in funding and this led to the first AI winter [16]. The high optimism had led to impossible expectations and the failure to deliver promised results led to a cut-off in AI funding.

In the 1980s, AI saw a surge of funding [16]. There was a hope for the revival of the field again. Few initial successes generated a new wave of investment from corporations around the world who were in pursuit of developing their own 'expert systems'. Things seemed to be hopeful once again with the renewed interests during the period of 1980-1987, where AI achieved great success. Regardless of these achievements, the technology was unable to live up to the unrealistic expectations due to maintenance cost, difficulty in scaling and limited scope, leading to AI seeing the second winter as funding got pulled back once again.

With increased computational power and huge amounts of electronic data being generated all across industry, the third surge of AI was seen in the 1990s [16, 23]. AI started to achieve great success, some being Google's search engine, data mining, speech recognition, medical diagnosis and so on. With Gary Kasparov being defeated in 1997 by Deep Blue from IBM, it became the first chess-playing computer to achieve such feat [48, 50]. During the early 21st century access to large amounts of data (big data) and advanced machine learning techniques, AI-related products have started to take over the market and all the possible big industries.

1.1 Machine Learning (ML)

This sub-branch of AI was born from the idea that machines can be trained for recognizing patterns and act independently with minimum human intervention through the analysis of big data. Statistics and mathematical optimization work together to form the backbone of machine learning and the easier and cheaper availability of computers along with modern computational elements like cloud computing and open source software, ML has become easily accessible to the public.

An AI learns from a given dataset using a ML technique that designs a mathematical function to fit the dataset. To fit this function accurately, the ML method needs to keep an account for the difference between the actual value from the observations in the dataset and the predicted or estimated value given by the designed function. This difference is formulated as the error function for every algorithm. The ML method is successful when it has reduced the error to an acceptable level. However, if the error is high, the algorithm attempts to improve the function until it reaches the level of acceptable error. Optimization techniques are required for this process of reducing errors and are an integral part of a ML algorithm. Stochastic gradient descent (SGD) is one of the popular optimization techniques in the recent times[23]. The SGD method helps an AI to learn rapidly from a dataset as its performance is dependent on accessing a smaller subset of data points from the given big

dataset. The original idea for SGD was formulated by Robbins and Monro in 1951 [52]. Since then SGD has seen large scale applications in ML models like logistic regressions, support vector machines, natural language procession and image recognition [9]. In this Chapter 1 we look at the fundamental framework of the SGD algorithm and see how it can be modified to design the adaptive gradient descent method (AdaGrad), treated in Chapter 2, and the stochastic alternating least squares (SALS) method, discussed in Chapter 3.

Typically an ML problem starts with a dataset $\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$, where the objective is to find a reasonable function h from within a set of admissible functions that best fits this information. For each $i \in 1, 2, \cdots, N$ the vector x_i represents the input with the corresponding output y_i and we expect $h(x_i) \approx y_i$. The admissible set is chosen to ensure that the function h not just memorizes the current dataset, but also offers good approximation for instances that do not appear in the examples. Rote memorization is avoided by constructing a *prediction function* h that can generalize the concepts that may be learned from the dataset, hence predicting the outcome y_i accurately for each input x_i. There are multiple choices of h that are used in practice. Most commonly, h is parameterized by a design parameter vector $u \in U$, where U is a set of admissible parameters. To achieve a prediction model is to obtain the optimal values of u that give us a model function h that not just satisfies the training values of the dataset, but can output the estimated value of y for an un-accessed datapoint x from the dataset. To achieve the desired accuracy of this prediction, a metric is used to measure how far h is from actual function value y_i for each instance in the dataset, and call it the *cost* function or *loss* function. Depending on the problem, we may have different constructions of the loss function, some examples being the following:

- Quadratic loss: $f_i(u) = (h(x_i; u) - y_i)^2$
- Cross entropy loss: $f_i(u) = -(y_i \ln(h(x_i; u)) + (1 - y_i) \ln(1 - h(x_i; u))$

- 0-1 loss: $f_i(u) = \mathbb{I}(h(x_i; u) \neq y_i)$ where $\mathbb{I}(\mathbf{A}) = \begin{cases} 1, & \text{if } \mathbf{A} \text{ is true} \\ 0, & \text{otherwise} \end{cases}$

For the low-rank tensor decomposition problem discussed in Chapter 3, the prediction function h is a low-rank tensor approximation, the parameter u represents its components, and the loss function $f_i(u)$ is the Frobenius distance between h and an observed sample tensor $\mathcal{X}^{(i)}$.

In practice, a regularization term is added to the loss function to keep the norm of the parameter u in check. A regularization term as the name suggests, makes the objective function more regular by restricting the fitting parameters, either explicitly or through the use of a penalty term. While regularization introduces a bias in the parameter estimates, it adds to the well-posedness of the problem while often also reducing the chance of overfitting. For example, if we choose the quadratic loss, we can use the L_2-norm of the parameters as the regularization term, thus constructing the loss function,

$$f_i(u) = \frac{1}{2}(h(x_i; u) - y_i)^2 + \frac{\alpha}{2}\|u\|^2$$

where the constant α is adjusted to control the effect of regularization.

The objective of learning from the dataset is achieved through minimization of the expected value of this loss function over all examples, i.e.,

$$\min_{u \in U} F(u) = \frac{1}{N} \sum_{i=1}^{N} f_i(u). \tag{1.1}$$

More generally, the loss function may be a random mapping $f : U \times \Omega \to \mathbb{R}$ defined over a complete probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is the sample space, $\mathcal{F}$ denotes the σ-algebra, and $\mathbb{P}$ denotes an appropriate probability measure. In Chapter 2 for example, the loss function $f(u, \omega)$ represents the L_2-deviation of the trajectory of a sample state from a desired target trajectory, under a given control function u. In the more general case, the

optimization problem is formulated as

$$\min_{u \in U} F(u) = \mathbb{E}[f(u)] = \int_{\Omega} f(u, \omega) d\mathbb{P}(\omega).$$

In practice this expectation $\mathbb{E}[f(u)]$ is usually approximated through Monte Carlo sampling, giving rise to a problem such as (1.1).

1.2 Optimization

The algorithms from ML we discuss in this thesis belong to the class of *gradient descent* algorithms. As the name suggests, these algorithms are based on the fact that the steepest descent of a function at a point in its domain is in the direction of the negative gradient at that point. Considering vector u in Equation 1.1, starting with an initial guess u_0, the following iteration gets us close to an optimal solution u^* as minimizer for the function $F(u)$:

$$u_{j+1} = u_j - \eta_j \cdot \nabla F(u_j). \tag{1.2}$$

We hope that with enough iterations we can get u_j to be as close to u^* as we want. A crucial parameter for the above iteration step is the stepsize η_j which determines the learning rate, i.e., how far we move in the direction of the steepest descent. A significant amount of research work always goes towards determining an appropriate value of η_j. It can be a constant value or a value that evolves as the number of iterations the algorithm has performed. A wrong choice of the step-size can lead to a very slow convergence of the algorithm or over-shooting, causing us to miss the optimal solution or oscillating around it. In our discussions we will consider a convex function and hence assume that it attains an unique minimum.

With our goal of minimizing the risk function given by the Equation 1.1 we apply the gradient descent method to Equation 1.2, we obtain the following batch gradient descent

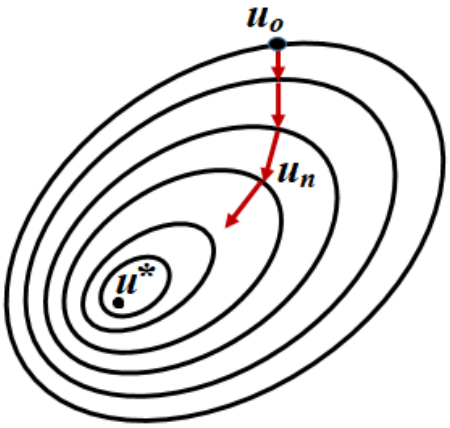


Figure 1.1: Contour plot showing the direction of the steepest descent

iteration step

$$u_{j+1} = u_j - \eta_j \cdot \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(u_j). \tag{1.3}$$

We observe that in order to obtain the gradient of the risk function for each iteration, we need to find the average of the gradient of the loss function over the entire dataset. This leads us to an extremely expensive computational process. For every single step of the iteration calculating the average over the entire dataset seems to be costly. Hence, we need to modify the algorithm so that it can still obtain accuracy by using a much smaller subset. In the following section we introduce the stochastic gradient descent method that achieves the above goal.

1.3 Stochastic Gradient Descent

Optimization methods in ML fall broadly under two categories, *stochastic* and *batch* gradient descent [9]. Discussed so far was the batch method which definitely without doubt has it's own merits. Using the batch approach we can utilize the already available deterministic gradient-based methods including non-linear optimization techniques that have been developed in the past decades. Due to the risk function being a sum, a batch process can easily be subjected to parallel computing over a distributed system. Regardless, stochastic

gradient descent algorithms have gained a lot of popularity in the recent times. In this thesis we mainly focus on SGD along with one of its variations, AdaGrad methods.

On large-scale applications involving big data, there are often redundancies, as copies of the same data in the dataset lead to repetition of information. When working with batch gradient descent, redundant data that provide no new information are taken into consideration when calculating the gradient and hence increasing computational cost. In comparison, the stochastic gradient descent method (SGD), follows the iteration steps with initital guess u_0,

$$u_{j+1} = u_j - \eta_j \cdot \nabla f_{i_j}(u) \tag{1.4}$$

which estimates the true gradient by only one instance f_{i_j} instead of ∇F and hence earning its name. The index i_j is randomly chosen from i.i.d on the set of indices $1, 2, \cdots, N$. The advantage of this process is that for every iteration, the computation of this estimate becomes significantly cheaper due to it being just one instance and not an average of gradients over a large dataset. The learning rate determined by the stepsize η_j is usually chosen as a constant or a diminishing stepsize $\eta_j = \frac{\eta}{j}$, with $\eta > 0$. Due to the stochastic nature of this algorithm we can always expect a speedy convergence as instances of the function picked can help the iteration jump close to the optimal solution quickly.

In pseudocode, we can observe the SGD method with diminishing stepsize as the following:

Algorithm 1 SGD method

1: Initialize u_0, η
2: **for** $j = 0, 1, \ldots$ **do**
3: Choose an instance given by index i_j
4: Compute $\nabla f_{i_j}(u_j)$
5: $u_{j+1} = u_j - \frac{\eta}{j} \nabla f_{i_j}(u_j)$
6: **end for**

In practice the learning rate has to be adjusted with a suitable choice of η for the efficient convergence. The diminishing stepsize usually gives a smoother convergence as the learning rate slows down with the passing of iterations hoping to get closer to the optimal minimizer. However, this usually does not take into account the amount of descent at the current point of iteration determined by the gradient. The idea of taking into account the gradient of the function for the evolution of the stepsize provides us the motivation for the Adaptive Gradient descent or AdaGrad method that we discuss in Chapter 2. In our pursuit of minimizing the risk function we are searching for the solution to the gradient of risk function being zero. The solution to this equation can be obtained using Newton's method which provides a fast convergence. Considering $H_j = \nabla^2 F(u)$, the Hessian of the Risk function, the iteration takes the form,

$$u_{j+1} = u_j - \eta H_j^{-1} \nabla F(u_j) \tag{1.5}$$

The computation of this Hessian can be expensive leading to a trade-off between the convergence rate and the cost of computation. We can instead estimate this Hessian by a simpler preconditioner matrix which usually is a positive semi-definite matrix for convex optimization. One idea would be to choose a diagonal matrix $B_j = B(u_j)$ to obtain the iteration step,

$$u_{j+1} = u_j - \eta_j B_j^{-1} \nabla F(u_j).$$

However, the normalizing effect of this preconditioner B_j can be further simulated by replacing with a scalar b_j demonstrated in [60], by the following iteration,

$$u_{j+1} = u_j - \eta \cdot \frac{1}{\sqrt{b_{j-1} + \|\nabla f_{i_j}(u_j)\|^2}} \cdot \nabla f_{i_j}(u_j)$$

The scalar b_j accounts for all the norm of the gradient values for the previous iterations. Thus b_j is a diminishing stepsize keeping into account the degree of descent of the function. To

obtain the stochastic algorithm once again the gradient of the risk function ∇F is estimated by the instance f_{i_j}.

In context to the design of optimal control of a stochastic partial differential equation (SPDE) in Chapter 2, the objective is to introduce a control function to the system to drive the solution towards a desired solution. The stochasticity of the SPDE is described by the random coefficients of the system, i.e. the coefficients are functions of a random state $\omega \in \Omega$ shown in Equation 2.1. The solution to the SPDE thus depends on the control u and the random state ω. Each instance of ω corresponds to an observation of the system. For each of these instances the loss is a function that calculates the difference of the solution of the SPDE from the desired solution, making it a function of u and ω. We consider the risk function $\mathbb{E}[f(u,\omega)]$ that needs to be minimized. We can consider our dataset as different instances of this system, thus generating the collection of loss functions $\{f_1(u), f_2(u), \cdots f_N(u)\}$ corresponding to each instance of ω over N-iterations where $f_j(u) = f(u,\omega_j)$. The gradient of the risk function $\mathbb{E}[\nabla f(u)]$ is estimated by one single instance $\nabla f(u,\omega)$ where the iteration takes the form

$$u_{j+1} = u_j - \underbrace{\frac{\eta}{b_j}}_{\text{learning rate}} \cdot \underbrace{\nabla f_j(u_j)}_{\text{gradient}} \quad \text{where} \quad b_{j+1}^2 = b_j^2 + \|\nabla f_j(u_j)\|^2$$

and the underbraces show the equivalent learning rate and estimated gradient from the Equation 1.4.

Chapter 3 demonstrates the stochastic alternative least Square (SALS). The objective of the SALS algorithm is to decompose a random tensor (tensor with random entries) as a sum of simpler tensors. Tensors are multidimensional arrays that are higher dimensional generalizations to vectors and matrices that appear very frequently in ML methods. Current methods of training neural networks heavily depend on tensors as they can represent the input, output and the transformations within the network as one big array of numbers. Being

a large array of numbers, a tensor can be computationally expensive to deal with. Decomposing a tensor as a sum of simpler tensors can provide us with a form that can accurately approximate the principle properties of the tensor and simultaneously provide a computationally cheaper array to deal with. To be more specific, in context to our experiments when we say simpler tensors, we mean rank-one tensors. A tensor that can be represented as an outer product of vectors is a rank-one tensor. Our objective is to approximate a random tensor χ by an approximate tensor $\tilde{\chi}$ that is a sum of rank-one tensors where the number of terms in the sum, r is predetermined and is defined as the rank of $\tilde{\chi}$. The loss function in the context to this problem measures the norm of the difference of $\chi - \tilde{\chi}$ with the regularization term as the square of the norm of $\tilde{\chi}$. Expected value of this loss function is defined as the risk function for the problem. To be able to apply the above algorithm to this problem, we vectorize the tensors in the risk function. If we observe the vectorized form of these tensors componentwise (that is, the vectors are sub-divided into smaller vectors called the components), we see that the risk function turns out to be quadratic for each component. This allows us to apply a least square fit succesively for each component. Specifically, in Chapter 3 we propose the stochastic iteration of the form

$$u_i^{k+1} = u_i^k - \underbrace{\alpha^{k,i}\left(H^{k,i}\right)^{-1}}_{\text{learning rate}} \cdot \underbrace{\tilde{\boldsymbol{g}}^{k,i}}_{\text{gradient}}$$

where

$$\alpha^{k,i} = \frac{c^{k,i}}{k}, \qquad \text{for } i = 1, 2, \ldots, p, \text{ and } k = 1, 2, \ldots,$$

u_i^k represents the kth parameter iterate in the ith component, $\tilde{\boldsymbol{g}}^{k,i}$ is the associated componentwise sample gradient, and $H^{k,i}$ is the componentwise Hessian of the loss. The underbraces show the equivalent learning rate and estimated gradient from the Equation 1.4. Following the index i we traverse the components of the vectorized form of the tensor and following index k we traverse the iteration steps. For each k-th iteration we need to traverse through

all the components of the vectorized form, thus exhausting the values of the index i from the set $\{1, 2, \cdots, p\}$ where p is the number of components.

Remark 1. *In Chapter 3, we will denote the parameter representing the rank-one tensor components by* $\boldsymbol{x}$*, in keeping with prevailing conventions in the literature on tensor decomposition.*

Chapter 2

Optimal Control

The work in this chapter appears in the paper *Adaptive Gradient Descent for Optimal Control of Parabolic Equations with Random Parameters*. Cao, Y., Das, S., van Wyk, H.-W., 2021. [12]

Stochastic optimization algorithms have increasingly found a use in the design of deterministic regulators for uncertain systems. Such problems arise in open loop control systems exhibiting statistical variations that cannot be observed by the controller. The control must consequently be designed to be robust in light of predicted uncertainties to ensure a desired statistical behavior of the system, as encoded by an appropriate risk function. In this chapter we measure risk in terms of the mean squared deviation of the controlled state from a desired reference state, but other risk functions, such as Conditional Value at Risk [53], are also possible.

We focus on the use of adaptive stochastic gradient methods in the distributed control of uncertain parabolic systems. These problems arise naturally in the thermal regulation of lithium battery systems [59, 25], for example. The variability of operating conditions, of the manufacturing process, and of the degradation of batteries over their life cycle gives rise to uncertainties in thermal properties of such systems. Their effect on material properties such as local resistivity, can be quantified through various methods, including empirical testing, and the use of battery degradation models [62]. The optimal controller is designed to ensure a desired average overall temperature over a range of conditions.

In mathematical terms, the design of an optimal control for uncertain systems can be formulated as a deterministic, infinite dimensional optimization problem whose cost function takes the form of a stochastic integral. Stochastic gradient (SG) algorithms are line search

methods in which the gradient of the risk function is replaced by random gradient samples, resulting in iterations that are much cheaper to compute than those obtained by a full approximation of the gradient. These stochastic optimization methods, originally conceived in [52] and fundamental in the development machine learning algorithms, have recently garnered attention in the context of infinite dimensional optimization (see e.g. [41, 21]). They are particularly well-suited to large scale problems in which the risk function is strongly convex and the underlying uncertainty is sufficiently complex to warrant the use of Monte Carlo sampling in approximating the stochastic integrals. It can be shown (see e.g. [9]) that for strongly convex risk functions whose sample gradients are Lipschitz continuous with bounded variance, the stochastic gradient method with appropriately chosen step-sizes converges at the rate $O(1/j)$, where j is the number of gradient evaluations. The convergence rate is optimal among first order methods, according to the complexity bounds established in [2]. In comparison, a gradient-based deterministic optimization scheme, coupled with standard Monte Carlo approximation, has a convergence rate of $O(1/\sqrt{j})$. The cost-reduction offered by SG iterations is especially pertinent in the context of optimal control, where the evaluation of the sample gradient involves the numerical approximation of two partial differential equations, namely the state and the adjoint equations (see Section 2.1).

While the SG iteration converges for a range of predetermined stepsizes, its convergence rate can vary widely. Moreover, the stepsize rule that guarantees the optimal convergence rate depends on the gradient's Lipschitz constant, its variance, and its strong convexity parameter, all of which are difficult to estimate in general. This practical shortcoming has led to investigations into adaptive stepsize rules that use information obtained during the iteration to adjust the stepsizes on the fly. The Adaptive Gradient (AdaGrad) method, developed concurrently in [17] and [43], scales the stepsize by the cumulative sum of gradients sampled thus far. Despite the widespread use of AdaGrad and its extensions, such as the Root Mean Square Propagation (RMSProp), or the Adaptive Moment Estimation (Adam) algorithms [28, 32], theoretical insight into its robustness and convergence has remained

elusive until recently [60, 37], even in the finite dimensional case. In [37], the authors prove the convergence of the AdaGrad method for both strongly convex and general functions over finite dimensional parameter spaces, obtaining convergence rates in the expected optimality gap that interpolate between $1/j$ and $1/\sqrt{j}$. In their proof the authors require a bound on the initial stepsize parameter that involves the risk function's Lipschitz constant. In [60], the authors prove a weaker form of convergence for a general risk function without imposing any conditions on the stepsize in terms of the underlying problem parameters.

This chapter aims to extend the AdaGrad method to infinite dimensional distributed control systems constrained by parabolic PDEs with uncertainties. In Section 2.1 we outline the optimal control problem and introduce the AdaGrad algorithm. In Section 3.3 we establish convergence of the AdaGrad algorithm and relate its requirements on the risk function's regularity, convexity, and finite variance to verifiable properties of the system's uncertain parameters. The numerical experiments in Section 2.3 illustrate the algorithm's theoretical properties derived in previous section, as well its appplication to a thermal regulation problem. In Section 3.5, we offer concluding remarks.

2.1 Problem Setting

We consider the non-homogeneous heat equation [19, 30] to model our system. Consider $(\Omega, \mathcal{F}, \mathbb{P})$ be a complete probability space encoding the uncertainties in our system, $D \subset \mathbb{R}^d$, $d = 1, 2, 3, \cdots$ be a physical domain with boundary ∂D, and $T > 0$ be some terminal time. The system's state $y : D \times [0, T] \times \Omega \to \mathbb{R}$ is then defined as the random field satisfying

$$\begin{cases} \frac{dy}{dt} - \nabla(a\nabla y) = g + u, & x \in D,\ t \in [0, T], \\ y = 0, & x \in \partial D,\ t \in [0, T], \\ y(\cdot, 0) = y_0, & x \in D \end{cases} \tag{2.1}$$

almost surely (a.s.) on Ω. For convenience, we define the differential operator $\mathcal{L} = \frac{d}{dt} - \nabla(a\nabla\cdot)$. Uncertainties in the system can arise from the diffusion coefficient a, the forcing term g, or the initial condition y_0. The deterministic function $u \in U := L^2(D \times [0,T])$ represents the distributed control to be determined through optimization. To ensure the state equation's well-posedness, we make the following assumptions.

Assumption 1. *The diffusion coefficient $a \in L^\infty([0,T] \times D \times \Omega)$ satisfies the coercivity condition*

$$0 < a_{\min} \leq a(x,t,\omega) \leq a_{\max} < \infty, \qquad x \in D,\ t \in [0,T],\ \omega \in \Omega. \tag{2.2}$$

for constants $a_{\min}, a_{\max} \in \mathbb{R}$. The forcing term g is a square integrable mapping $g : \Omega \to L^2([0,T], H^{-1}(D))$, i.e. $g \in L^2(\Omega, L^2([0,T], H^{-1}(D))$, and the initial condition $y_0 \in L^2(D \times \Omega)$.

As a consequence, Equation (2.1) has a unique solution y for every control u [5]. The assumption on the uniform coercivity can be relaxed somewhat to allow for lognormal diffusion coefficients, see [22]. In the following, we will often find it useful to refer to y in terms of its dependence on various subsets of x, t, ω, and the control u, e.g. $y(u,\omega)$ or $y(x,t,\omega)$. Throughout the paper, we will use $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ to refer to the inner product and norm associated with $L^2([0,T] \times D)$, i.e. for $v \in L^2([0,T] \times D)$,

$$\|v\|^2 = \int_0^T \int_D |v(x,t)|^2 dx\, dt.$$

2.1.1 The Optimal Control Problem

In optimal control we seek a controller u that steers the corresponding state $y(u)$ to track a desired reference solution y_d. This function is often deterministic, but here we need only require $y_d \in L^2([0,T] \times D \times \Omega)$, allowing us to encode the state's desired behavior on a statistical level. We measure the deviation of $y(u)$ from y_d in a mean-squared sense

over all possible state realizations. Furthermore, we constrain the control u to lie within the admissible set

$$U_{ad} = \{u \in U : \|u\| \leq u_{\max}\},$$

for a given constant $0 < u_{\max} \leq \infty$. The optimal control problem can thus be stated as

$$\min_{u \in U_{ad}} F(u), \tag{2.3}$$

where the risk function is

$$F(u) = \mathbb{E}[f(u)] = \int_{\Omega} f(u, \omega) d\mathbb{P}(\omega),$$

and

$$f(u, \omega) = \frac{1}{2}\|y(u, \omega) - y_d(\omega)\|^2 + \frac{\alpha}{2}\|u\|^2.$$

To establish well-posedness of Problem (2.3), i.e. the existence and uniqueness of an optimal control, we note that the risk function F can be readily decomposed into

$$F(u) = \pi(u, u) - 2L(u) + C,$$

where

$$\pi(u, v) = \frac{1}{2}\mathbb{E}\left[\langle y(u) - y_0, y(v) - y_0\rangle\right] + \frac{\alpha}{2}\mathbb{E}\left[\langle u, v\rangle\right]$$

is a continuous, coercive bilinear form,

$$L(u) = \mathbb{E}\left[\langle y_d - y_0, y(u) - y_0\rangle\right], \text{ and}$$

is a bounded linear form, C is constant in u. The result then follows directly from Chapter 1, Theorem 1.1 in [38]. In fact, it will be shown in 2.2.1 that $f(u, \omega)$ is strongly convex

a.s. on Ω, which has strong implications on the convergence rate of the AdaGrad method introduced in Section 2.1.2.

2.1.2 The AdaGrad Algorithm

While Problem (2.3) is ostensibly deterministic, its risk function $F(u)$ is a statistic associated with the uncertain system (2.1). For a given initial guess u_0, consider the stochastic gradient iteration

$$u_{j+1} = u_j - \eta_j \nabla f_j(u_j), \tag{2.4}$$

where the stochastic gradients $\{\nabla f_j(u_j)\}_{j=1}^{\infty}$ are idependent, identically distributed samples of $\nabla f(u_j, \omega)$ satisfying $\mathbb{E}[\nabla f_j(u_j)] = \nabla F(u_j)$, and $\{\eta_j\}_{j=1}^{\infty}$ is a sequence of positive stepsizes.

Remark 2. *In cases when the risk function is determined empirically, i.e. by sampling, when the system's statistical distribution is determined empirically, through analysis of test samples, the risk function may take the form*

$$F(u) = \frac{1}{N} \sum_{i=1}^{N} f_i(u).$$

In this case, sample gradients can be obtained as $\nabla f_{i_j}(u)$, where i_j is drawn uniformly from the set $\{1, \ldots, N\}$ (see [9]).

The established convergence analysis of the SGD algorithm for non-convex smooth functions depends on specific conditions on positive step-size η_j (see [52]). A sufficient condition for convergence is that $\{\eta_j\}_{j=1}^{\infty}$ is a deterministic sequence of positive numbers that satisfies:

$$\sum_{j=1}^{\infty} \eta_j = \infty \quad \text{and} \quad \sum_{j=1}^{\infty} \eta_j^2 < \infty \tag{2.5}$$

The choice of η_j can be a conveniently chosen constant or diminishing function of the iteration count, i.e., $\eta_j = O(\frac{1}{j})$. As an improvement to this SGD algorithm we apply the

Adaptive Gradient Descent algorithm that uses the following adaptive step-size,

$$\eta_j = \frac{\eta}{\sqrt{b_0^2 + \sum_{k=1}^{j-1} \|\nabla f_k(u_k)\|^2}} \tag{2.6}$$

where $\eta > 0$ and $b_0 > 0$ are constants. Other versions of AdaGrad scale componentwise. Other variations include the current gradient value (though this can lead to steps that are not descent directions in expectation [37]).

For the optimal control problem (2.3), it is a standard result (see [38]) that $f(u)$ is Fréchet differentiable and that the sample gradients $\nabla f(u)$ can be computed as

$$\nabla f(u) = p + \alpha u, \tag{2.7}$$

where p is the adjoint state, satisfying

$$\begin{cases} -\frac{dp}{dt} - \nabla \cdot (a \nabla p) = y - y_d, & x \in D,\ t \in [0,T], \\ p(x,t) = 0, & x \in \partial D,\ t \in [0,T], \\ p(x,T) = 0, & x \in D. \end{cases} \tag{2.8}$$

Indeed, the directional derivative $D[f(u)](v)$ of f at $u \in U$ in any direction $v \in U$ is given by

$$D[f(u)](v) = \langle y - y_d, s(v) \rangle + \alpha \langle u, v \rangle,$$

where $s(v)$ satisfies the sensitivity equations

$$\begin{cases} \frac{ds}{dt} - \nabla \cdot (a \nabla s) = v, & \forall x \in D,\ t \in [0,T], \\ s(x,t) = 0, & \forall x \in \partial D,\ t \in [0,T], \\ s(x,0) = 0, & \forall x \in D. \end{cases} \tag{2.9}$$

Denoting by $\mathcal{L}^* = -\frac{d}{dt} - \nabla(a\nabla\cdot)$ the formal adjoint of $\mathcal{L}$, we can use Equation (2.8), integration by parts, and Equation (2.9) to write

$$\begin{aligned} D[f(u)](v) &= \langle y - y_d, s(v)\rangle + \alpha\langle u, v\rangle = \langle \mathcal{L}^* p, s(v)\rangle + \alpha\langle u, v\rangle \\ &= \langle p, \mathcal{L}s(v)\rangle + \alpha\langle u, v\rangle = \langle p, v\rangle + \alpha\langle u, v\rangle, \end{aligned}$$

from which Formula (2.7) follows.

The AdaGrad algorithm for Problem (2.3) can thus be summarized as follows.

Algorithm 2 AdaGrad for Problem (2.3)

1: Initialize u_0, b_0, η
2: **for** $j = 0, 1, \ldots$ **do**
3: $\quad y_j \leftarrow$ solution of sample primal system (2.1)
4: $\quad p_j \leftarrow$ solution of sample adjoint system (2.8)
5: $\quad \nabla f_j(u_j) = p_j + \alpha u_j$
6: $\quad u_{j+1} = u_j - \frac{\eta}{b_j}\nabla f_j(u_j)$
7: $\quad b_{j+1}^2 = b_j^2 + \|\nabla f_j(u_j)\|^2$
8: **end for**

2.2 Convergence Analysis

In this section we establish theoretical properties of the AdaGrad algorithm in the context of the optimal control problem (2.3). Theorem 2.1 proves AdaGrad's convergence under suitable conditions. To this end we show that the problem's risk function is strongly convex (Proposition 2.2.1) and Lipschitz continuous (Proposition 2.2.2). Moreover, in Section 2.2.2 we relate the algorithm's finite variance requirement to the statistical properties of the underlying uncertain inputs. Our convergence analysis is based on that of [37] whose authors establish convergence for convex risk functions over finite-dimensional parameter

space, whose sample gradient functions ∇f are Lipschitz continuous, unbiased, and uniformly bounded over u and $\omega \in \Omega$.

2.2.1 Convexity and Smoothness

Proposition 2.2.1. *The mapping $f(\cdot, \omega) : U \to \mathbb{R}$ is strongly convex a.s. on Ω. Specifically,*

$$\langle \nabla f(v, \omega) - \nabla f(u, \omega), v - u \rangle \geq \alpha \|v - u\|^2 \qquad \text{a.s. on } \Omega. \tag{2.10}$$

Proof. Since we will show that the result holds a.s. on Ω, we find it notationally convenient not to write ω explicitly in this proof. Specifically, let $\omega \in \Omega$ be a fixed system realization and let $y(u)$, $p(u)$, and $f(u)$ be respectively the associated state, adjoint variable, and sample cost when subjected to control $u \in U$. Using the form of the gradient given by Equation (2.7), we have that for $u, v \in U$,

$$\langle \nabla f(v) - \nabla f(u), v - u \rangle = \alpha \|v - u\|^2 + \langle p(v) - p(u), v - u \rangle. \tag{2.11}$$

To prove the proposition, it therefore suffices to show $\langle p(v) - p(u), v - u \rangle \geq 0$. To this end, note that subtracting Equation (2.1) with control u from that with control v gives

$$\mathcal{L}(y(v) - y(u)) = v - u. \tag{2.12}$$

Similarly,

$$\mathcal{L}^*(p(v) - p(u)) = y(v) - y(u). \tag{2.13}$$

Multiplying Equation (2.12) by $p(v) - p(u)$ and integrating over D and $[0, T]$ gives

$$\langle \mathcal{L}(y(v) - y(u)), p(v) - p(u) \rangle = \langle v - u, p(v) - p(u) \rangle,$$

which, through integration by parts and Equation (2.13), takes the form

$$\begin{aligned}\langle p(v) - p(u), v - u\rangle &= \langle \mathcal{L}(y(v) - y(u)), p(v) - p(u)\rangle \\ &= \langle y(v) - y(u), \mathcal{L}^*(p(v) - p(u))\rangle = \|y(v) - y(u)\|^2 \geq 0.\end{aligned}$$

This proves Inequality (2.10), by virtue of Equation (2.11). □

Proposition 2.2.2. *For f defined in (2.7), $u, v \in U$, and for $\omega \in \Omega$, we have*

$$\|\nabla f(v) - \nabla f(u)\| \leq M\|v - u\|, \tag{2.14}$$

where $M = \alpha + \frac{C_p^4}{a_{\min}^2}$.

Proof. As before we refrain from writing f, y, and p explicitly in terms of the random state $\omega \in \Omega$. Recall Equation (2.11)

$$\nabla f(v) - \nabla f(u) = \alpha(v - u) + p(v) - p(u),$$

for a given $\omega \in \Omega$, and controls $u, v \in U$. To bound the difference in sample gradients, it therefore suffices to bound the difference in adjoint variables. Multiplying Equation (2.13) by $p(v) - p(u)$, integrating over D and $[0, T]$, and using Green's theorem for the diffusion term, as well as the chain rule for the time derivative yields

$$\begin{aligned}&\langle y(v) - y(u), p(v) - p(u)\rangle = \langle \mathcal{L}^*(p(v) - p(u)), p(v) - p(u)\rangle \\ &= -\int_0^T \int_D \frac{d}{dt}\|p(v) - p(u)\|^2 dx\, dt + \int_0^T \int_D a(\nabla p(v) - \nabla(p(u))^2 dx\, dt.\end{aligned} \tag{2.15}$$

The fundamental theorem of calculus and the terminal condition for the adjoint Equation (2.8) together imply

$$-\int_0^T \int_D \frac{d}{dt}\|p(x,t,v) - p(x,t,u)\|^2 dx\, dt = \int_D \|p(x,0,v) - p(x,0,u)\|^2 dx \geq 0.$$

Moreover, the coercivity condition (2.2), used in conjunction with the Poincaré inequality, implies

$$\int_0^T \int_D a(\nabla p(v) - \nabla(p(u))^2 dx\, dt \geq a_{\min} \|\nabla p(v) - \nabla p(u)\|^2 \geq \frac{a_{\min}}{C_p^2} \|p(v) - p(u)\|^2,$$

where C_p is the appropriate Poincaré constant. Equation (2.15) and the Cauchy-Schwartz inequality thus lead to

$$\frac{a_{\min}}{C_p^2} \|p(v) - p(u)\|^2 \leq \langle y(v) - y(u), p(v) - p(u) \rangle \leq \|y(v) - y(u)\| \|p(v) - p(u)\|,$$

and hence

$$\|p(v) - p(u)\| \leq \frac{C_p^2}{a_{\min}} \|y(v) - y(u)\|. \tag{2.16}$$

By applying analogous arguments to the state Equation (2.1), we can similarly bound

$$\|y(v) - y(u)\| \leq \frac{C_p^2}{a_{\min}} \|v - u\|,$$

so that

$$\|p(v) - p(u)\| \leq \frac{C_p^4}{a_{\min}^2} \|v - u\|.$$

Therefore

$$\|\nabla f(v) - \nabla f(u)\| \leq \|p(v) - p(u)\| + \alpha \|v - u\| \leq \frac{C_p^4}{a_{\min}^2} \|v - u\| + \alpha \|v - u\|,$$

yielding the bound (2.14). □

2.2.2 Finite Variance

In addition to strong convexity and Lipschitz continuity, shown in Propositions 2.2.1 and 2.2.2 respectively, our convergence proof of the AdaGrad algorithm requires a strengthened finite variance condition on the sample gradients. It is commonly enforced [46, 37] by

assuming there is a constant $\sigma^2 > 0$ so that

$$\mathbb{E}\left[\exp\left(\frac{\|\nabla f(u) - \nabla F(u)\|^2}{\sigma^2}\right)\right] \leq \exp(1), \qquad \text{for all } u \in U. \tag{2.17}$$

Through the use of Jensen's inequality and conditional expectation, Inequality (2.17) can be readily shown (see e.g. [37]) to imply

$$\mathbb{E}\left[\max_{1\leq k\leq j} \|\nabla f_k(u_k) - \nabla F(u_k)\|^2\right] \leq \sigma^2(1 + \ln(j)), \qquad j = 1, 2, \ldots, \tag{2.18}$$

in addition to the standard variance bound

$$\mathbb{E}\left[\|\nabla f_j(u_j) - \nabla F(u_j)\|^2\right] \leq \sigma^2, \qquad j = 1, 2, \ldots. \tag{2.19}$$

Finite variance assumptions, such as Inequality (2.17), are commonly made in convergence analyses of stochastic optimization methods and are, strictly speaking, only required to hold for the iterations u_j. Nevertheless, they are generally not verifiable independently of the iteration. The following proposition helps frame requirement (2.17) in terms of statistical paramaters underlying the control problem (2.3).

Proposition 2.2.3. *For any $\omega \in \Omega$ and any fixed $u \in U$,*

$$\|\nabla f(u,\omega)\| \;\leq\; \left(\alpha + \frac{C_p^4}{a_{\min}^2}\right)\|u\| \;+\; \frac{C_p^4}{a_{\min}^2}\|g(\omega)\| \;+\; \frac{C_p^4}{2a_{\min}^2}\|y_0(\omega)\| \;+\; \frac{C_p^2}{a_{\min}}\|y_d(\omega)\|. \tag{2.20}$$

Proof. Recall from Equation (2.7) that

$$\|\nabla f(u,\omega)\| = \|\alpha u + p(u,\omega)\| \leq \alpha\|u\| + \|p(u,\omega)\|,$$

for any $\omega \in \Omega$, where $p = p(u,\omega)$ satisfies the adjoint system given by Equation (2.8). To bound $\|p\|$, we multiply on both sides of Equation (2.8) by p and integrate over $[0, T]$ and

D, leading to

$$-\int_0^T \int_D p_t p \, dx \, dt - \int_0^T \int_D \nabla \cdot (a \nabla p) p \, dx \, dt = \int_0^T \int_D (y - y_d) p \, dx \, dt. \tag{2.21}$$

Using the chain rule, the fundamental theorem of calculus, and the terminal condition for p, we obtain

$$-\int_0^T \int_D p_t p \, dx \, dt = -\int_0^T \int_D \frac{1}{2} \frac{d}{dt} p^2 \, dx \, dt = -\frac{1}{2} \int_D p(x,0)^2 dx \, dt \leq 0. \tag{2.22}$$

Moreover, by Green's theorem

$$-\int_0^T \int_D \nabla \cdot (a \nabla p) p \, dx \, dt = \int_0^T \int_D a \nabla p \cdot \nabla p \, dx \, dt \geq a_{\min} \|\nabla p\|^2. \tag{2.23}$$

Substituting Equations (2.22) and (2.23) into (2.21), rearranging terms, and using the Poincaré inequality then results in

$$\|p\| \leq \frac{C_p^2}{a_{\min}} \|y - y_d\| \leq \frac{C_p^2}{a_{\min}} \left(\|y\| + \|y_d\| \right). \tag{2.24}$$

To bound $\|y\|$, we multiply the state Equation (2.1) by y on both sides and use analogous arguments to those above to obtain

$$a_{\min} \|\nabla y\|^2 \leq (\|g\| + \|u\|) \|y\| + \frac{1}{2} \|y_0\|^2.$$

Since

$$\|y_0\| = \sqrt{\int_D y_0(x)^2 \, dx} \leq \sqrt{\int_0^T \int_D y(x,t)^2 dx \, dt} = \|y\|,$$

we have, by virtue of Poincaré's inequality, that

$$\|y\| \leq \frac{C_p^2}{a_{\min}} \left(\|g\| + \|u\| + \frac{1}{2} \|y_0\| \right).$$

Substituting this inequality into (2.24) establishes the estimate in (2.20). □

Note that the upper bound (2.20) involves the norm $\|u\|$ of the control. In fact, a derivation similar to that used to establish Proposition 2.2.3 leads to an upper bound for $\|\nabla f(u,\omega) - \nabla F(u)\|$, for $\omega \in \Omega$, that also depends on $\|u\|$, as long as the diffusivity a is stochastic. We therefore find it necessary to restrict the admissible set U_{ad} to be bounded, i.e. $u_{\max} < \infty$. To ensure that AdaGrad iterations remain feasible, we make the following assumption.

Assumption 2. *Assume that the samples and stepsizes in the AdaGrad iteration are chosen so that $u_j \in U_{ad}$ for $j = 1, 2, \ldots$.*

This condition can be enforced for example by rejecting steps that lie outside U_{ad}. To be sure, $u_{\max}$ may be chosen large enough to make such interventions unlikely as well as to ensure that the optimal control u^* lies in the interior of U_{ad}. In fact, by the optimality of u^* and form of the risk function $F(u)$, we have

$$\frac{\alpha}{2}\|u^*\|^2 \leq F(u^*) \leq F(u_0),$$

so that $\|u^*\| < u_{\max}$ whenever $u_{\max} > \sqrt{\frac{2}{\alpha}F(u_0)}$ for any initial condition u_0.

We will use the estimates of Proposition 2.2.3 to establish conditions on the uncertain parameters that ensure the strengthened bounded variance condition (2.17). To simplify the bound (2.20), we estimate $\|u\|$ by $u_{\max}$ and bound the constants by their upper bound $K = \max\left\{\alpha + \frac{C_p^4}{a_{\min}^2}, \frac{C_p^2}{a_{\min}}\right\}$, yielding

$$\|\nabla f\| \leq K(u_{\max} + \|g\| + \|y_d\| + \|y_0\|). \tag{2.25}$$

For any $\omega \in \Omega$, repeated use of Jensen's inequality gives

$$\mathbb{E}\left[\exp\left(\frac{\|\nabla f(u,\omega) - \nabla F(u)\|^2}{\sigma^2}\right)\right] \leq \mathbb{E}\left[\exp\left(\frac{2}{\sigma^2}\left(\|\nabla f(u,\omega)\|^2 + \mathbb{E}\left[\|\nabla f(u)\|^2\right]\right)\right)\right],$$

where (2.25) and Assumption (1) imply

$$\mathbb{E}\left[\|\nabla f(u)\|^2\right] \leq 4K^2\mathbb{E}\left[u_{\max}^2 + \|g\|^2 + \|y_d\|^2 + \|y_0\|^2\right] < \infty.$$

In light of these bounds, we can now formulate an assumption on the system parameters g, y_d, and y_0 that, together with Assumptions 1 and 2, implies the strengthened bounded variance condition (2.17).

Assumption 3. *Assume that there exists a $\sigma^2 > 0$, so that*

$$\mathbb{E}\left[\exp\left(\frac{8K^2}{\sigma^2}\left(\|g\|^2 + \|y_d\|^2 + \|y_0\|^2 + \mathbb{E}\left[\|g\|^2 + \|y_d\|^2 + \|y_0\|^2\right]\right)\right)\right] \leq \exp\left(1 - \frac{16K^2}{\sigma^2}u_{\max}^2\right). \quad (2.26)$$

With the cost function f satisfying Propositions 2.1-2.4 and from **Theorem 3** in [37] we can state that:

Theorem 2.1. *For stepsizes given by* (2.6)*, where $\eta, b_0 > 0$ and $4\eta M < \sqrt{b_0}$, and under Assumptions 1–3, the iterates of the AdaGrad algorithm satisfy the following bound*

$$\mathbb{E}\left[\sqrt{F(\bar{u}_n) - F(u^*)}\right] \leq \frac{1}{\sqrt{n}}\max\left(\gamma\sqrt{M}, (b_0 + n\sigma^2)^{\frac{1}{4}}\sqrt{\gamma}\right), \quad (2.27)$$

where

$$\bar{u}_n = \frac{1}{n}\sum_{j=1}^{n} u_j, \qquad \text{and} \qquad \gamma = O\left(\frac{1 + \eta^2 \ln n}{\eta(1 - \frac{4\eta M}{\sqrt{b_0}})}\right).$$

Remark 3. *The convergence bound ensures, via Markov's inequality, that the optimality gap* $F(\bar{u}_n) - F(u^*)$ *will satisfy*

$$F(\bar{u}_n) - F(u^*) \leq \frac{1}{\delta}\left(\frac{1}{n}\max\left\{\gamma^2 M, (b_0 + \sigma^2 n)^{\frac{1}{2}}\gamma\right\}\right)$$

with a probability of at least $1 - \delta$*. This bound highlights the effect of the variance on the method's convergence rate, reducing it from* $O\left(\frac{1}{\sqrt{n}}\right)$ *to* $O\left(\frac{1}{n}\right)$*.*

Remark 4. *Note that, while the stepsize rule* $\{\eta_j\}_{j=1}^{\infty}$ *does not depend on the variance parameter* σ^2*, it is constrained by the Lipschitz constant* M*. In [60] the authors provide a proof for a weaker form of convergence without any such dependence.*

Proof. For completeness, we give a brief outline of the proof. A more detailed discussion can be found in [37]. Let $\delta F_j = F(u_j) - F(u^*) \geq 0$ for $j = 0, 1, 2, \ldots$ be the optimality gap at the jth iteration. The convexity of F implies, by virtue of Jensen's inequality, that

$$\begin{aligned}\mathbb{E}\left[\sqrt{F(\bar{u}_n) - F(u^*)}\right] &= \mathbb{E}\left[\sqrt{F\left(\frac{1}{n}\sum_{j=1}^{n} u_j\right) - F(u^*)}\right] \\ &\leq \mathbb{E}\left[\sqrt{\frac{1}{n}\sum_{j=1}^{n} F(u_j) - F(u^*)}\right] = \frac{1}{\sqrt{n}}\mathbb{E}\left[\sqrt{\sum_{j=1}^{n}\delta F_j}\right]\end{aligned}$$

Moreover, the fact that $\eta_0 \geq \eta_1 \ldots \geq \eta_n > 0$, together with Hölder's inequality, justify

$$\mathbb{E}\left[\sqrt{\sum_{j=1}^{n}\delta F_j}\right] \leq \mathbb{E}\left[\sqrt{\frac{1}{\eta_n}\sum_{j=1}^{n}\eta_j \delta F_j}\right] \leq \left(\mathbb{E}\left[\frac{1}{\eta_n}\right]\right)^{\frac{1}{2}}\left(\mathbb{E}\left[\sum_{j=1}^{n}\eta_j\delta F_j\right]\right)^{\frac{1}{2}}. \tag{2.28}$$

To complete the proof, it now remains to bound $\mathbb{E}\left[\frac{1}{\eta_n}\right]$ and $\mathbb{E}\left[\sum_{j=1}^n \eta_j \delta F_j\right]$. From the definition of the stepsize, we have

$$\begin{aligned}\frac{1}{\eta_n} &= \frac{1}{\eta}\sqrt{b_0^2 + \sum_{j=1}^{n-1} \|\nabla f_j(u_j)\|^2} \\ &\leq \frac{1}{\eta}\sqrt{b_0^2 + 2\sum_{j=1}^{n-1} (\|\nabla f_j(u_j) - \nabla F(u_j)\|^2 + \|\nabla F(u_j)\|^2)}.\end{aligned}$$

By Lipschitz continuity, $\|\nabla F(u_j)\|^2 \leq 2M\delta F_j$, while the finite variance condition (2.17) implies, via (2.19), that $\mathbb{E}\left[\sum_{j=1}^{n-1} \|\nabla f_j(u_j) - \nabla F(u_j)\|^2\right] \leq (n-1)\sigma^2$. Hence, since $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$,

$$\mathbb{E}\left[\frac{1}{\eta_n}\right] \leq \frac{1}{\eta}\left(\sqrt{b_0^2 + 2(n-1)\sigma^2} + 2\sqrt{M}\mathbb{E}\left[\sqrt{\sum_{j=1}^n \delta F_j}\right]\right) \tag{2.29}$$

The bound on $\mathbb{E}\left[\sum_{j=1}^n \eta_j \delta F_j\right]$ can be established, based on the relation

$$\eta_j \langle \nabla f_j, u_j - u^* \rangle = \frac{1}{2}\|u_{j+1} - u^*\|^2 - \frac{1}{2}\|u_j - u^*\|^2 + \eta_j^2 \|\nabla f_j\|^2,$$

which can be readily derived using the AdaGrad update $u_{j+1} = u_j - \eta_j \nabla f_j$. Let $\mathbb{E}_j$ denote the conditional expectation with respect to ξ_j, given $\xi_1, \ldots, \xi_{j-1}$. Strong convexity and the fact that u_j and η_j are independent of ξ_j now imply

$$\begin{aligned}\mathbb{E}_j\left[\eta_j \delta F_j\right] &\leq \eta_j \langle \nabla F_j(u_j), u_j - u^* \rangle = \mathbb{E}_j\left[\langle \eta_j \nabla f_j, u_j - u^* \rangle\right] \\ &= \mathbb{E}_j\left[\frac{1}{2}\|u_{j+1} - u^*\|^2 - \frac{1}{2}\|u_j - u^*\|^2 + \eta_j^2 \|\nabla f_j\|^2\right],\end{aligned}$$

which, by summing over $j = 1, \ldots, n$, taking total expectation and recognizing a telescoping sum, yields

$$\begin{aligned}\mathbb{E}\left[\sum_{j=1}^{n} \eta_j \delta F_j\right] &= \mathbb{E}\left[\sum_{j=1}^{n} \mathbb{E}_j\left[\eta_j \delta F_j\right]\right] \\ &\leq \mathbb{E}\left[\sum_{j=1}^{n}\left(\frac{1}{2}\|u_j - u^*\|^2 - \frac{1}{2}\|u_{j+1} - u^*\|^2 + \frac{1}{2}\eta_j^2\|\nabla f_j\|^2\right)\right] \\ &\leq \frac{1}{2}\|u_1 - u^*\|^2 + \frac{1}{2}\mathbb{E}\left[\sum_{j=1}^{n} \eta_j^2\|\nabla f_j\|^2\right]. \end{aligned} \tag{2.30}$$

An upper bound $\mathbb{E}\left[\sum_{j=1}^{n} \eta_j^2\|\nabla f_j\|^2\right]$ can be obtained from the strong finite variance condition (2.17), as well as Lipschitz continuity, giving

$$\begin{aligned}\frac{1}{2}\mathbb{E}\left[\sum_{j=1}^{n} \eta_j^2\|\nabla f_j\|^2\right] \leq & \frac{1}{1 - \frac{4\eta M}{\sqrt{b_0}}}\frac{2\eta^2}{b_0}(1 + \ln n)\sigma^2 \\ & + \frac{\eta^2}{1 - \frac{4\eta M}{\sqrt{b_0}}} \ln\left(\sqrt{b_0 + 2n\sigma^2} + 2\sqrt{M}\mathbb{E}\left[\sqrt{\sum_{j=1}^{n} \delta F_j}\right]\right).\end{aligned}$$

Substituing this inequality into (2.30) and then inequalities (2.30) and (2.29) into (2.28), one obtains an inequality in $\mathbb{E}\left[\sqrt{\sum_{j=1}^{n} \delta F_j}\right]$, which can be solved (see [37], Lemmas 5 and 6), yielding the result. □

2.3 Numerical Experiments

In this section we perform two numerical experiments to illustrate and showcase the properties of the AdaGrad method for the parabolic optimal control problem (2.3). Example 1 explores the convergence behavior of the algorithm and compares it with that of the traditional SGD method, focusing specifically on the effects of stepsize and convexity. Example 2 applies the method to a simplified thermal regulator problem aimed at maintaining a safe overall temperature

Example 1. We first consider the parabolic problem given by Equation (2.1) with spatial domain $[0,1]$ and terminal time $T = 0.2$. Let the initial condition be given by $y_0(x) = x(1-x)$, the forcing term be $g(x) = 0$, and the diffusion coefficient be the lognormal random field,

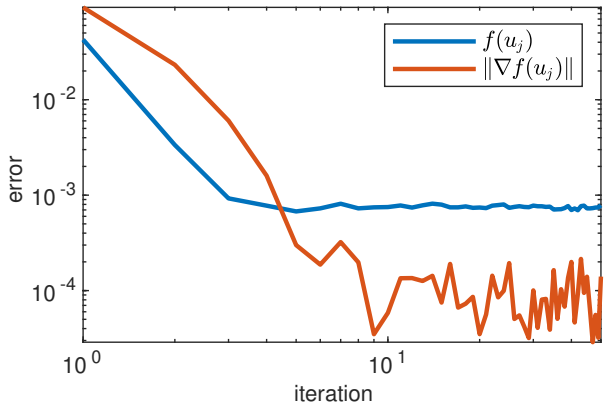
$$a(x,\omega) = a_{\min} + \exp(\tilde{a}(x,\omega)), \qquad x \in [0,1], \omega \in \Omega,$$

where $a_{\min} = 0.1$ and $\tilde{a}(x,\omega)$ is a zero-mean Gaussian random field with a Gaussian covariance kernel $k(x_1,x_2) = \sigma^2 \exp\left(-\frac{|x_1-x_2|^2}{2l^2}\right)$ with variance $\sigma^2 > 0$ and correlation length $l > 0$. In our experiments, the field is approximated by the truncated Karhunen-Loève expansion with 40 terms. The optimal control problem seeks to steer the state y to the zero state $y_d(x,t) = 0$ through distributed forcing with a regularization parameter $\alpha = 0.1$. Both the state and adjoint equations are approximated by piecewise linear finite elements with 50 sub-intervals in space, and the implicit Euler timestepping scheme with 100 sub-intervals in time.

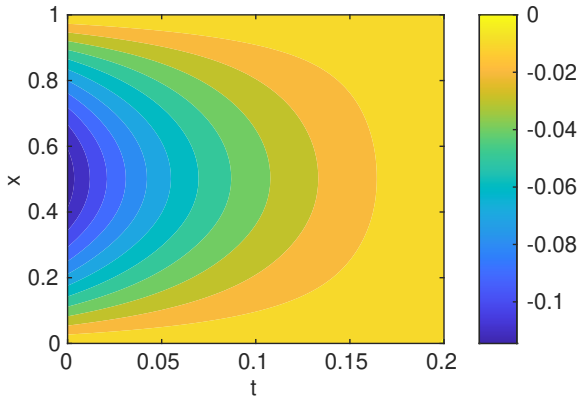
To verify that the AdaGrad method outlined by Algorithm 2 converges to an optimal control in mean square, we plot the sampled cost function $f(u_j,\omega)$ as well as its gradient $\nabla f(u_j,\omega)$ against the iteration count $j = 1,\dots 50$ in Figure 2.1. We use an initial guess of $u_0(x,t) = 2$ and optimization parameters $b_0 = 0.1$ and $\eta = 1$. As is usual in stochastic optimization, the cost initially decreases much variation during its transient phase (roughly 10 iteration steps here), after which it settles down into a stationary phase. Moreover, the linear decrease in f in the log-log scale during the transient phase suggests a convergence rate of $O(\frac{1}{j})$. In light of Remark 3, this is to be expected due to the small variance, as shown in Figure 2.2b.

In Figure 2.2 we plot the sample mean and variance of the state y under the optimal control u to verify that the desired state $y_d(x,t) = 0$ is tracked.

Next we compare the AdaGrad algorithm's convergence behavior with that of the SGD method. Figure 2.3 shows the effect of the stepsize on the algorithms' performances. The stepsize rule for the SGD method is chosen as $\eta_j = \frac{\eta_0}{j+1}$ for $j = 0,1,...200$, while that of the

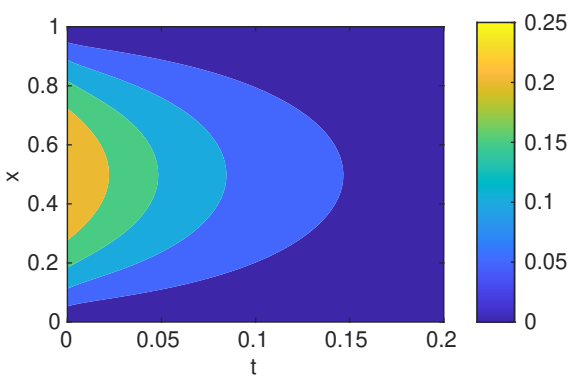


(a) Convergence of the cost and gradient norm on a log-log scale.

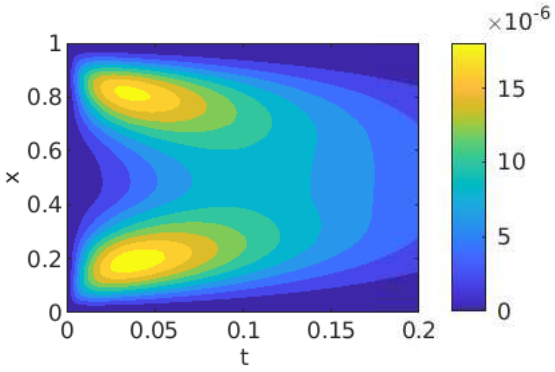


(b) Optimal control u

Figure 2.1: Convergence of the AdaGrad Method



(a) The sample mean of the optimal state y.



(b) The sample variance of the optimal state y.

Figure 2.2: Sample statistics of the optimal state using a sample size of 100.

AdaGrad algorithm was chosen according to Equation (2.6), with $b_0 = 1$ to ensure a fair comparison. Evidently, the stepsize affects the convergence behavior of both algorithms for this problem, with a decrease in η_0 leading to a deterioration in convergence. However, the AdaGrad method (in black) seems to be less adversely affected than the SGD method (in red).

Figure 2.4 shows how the regularization parameter α, which determines the problem's strong convexity, plays a role in the algorithms' convergence behaviors. For both algorithms we set the stepsize parameters to ensure an initial stepsize of $\eta_0 = 10$ and vary the regularization parameter α. The plot suggests that, while a reduction in the strong convexity of the

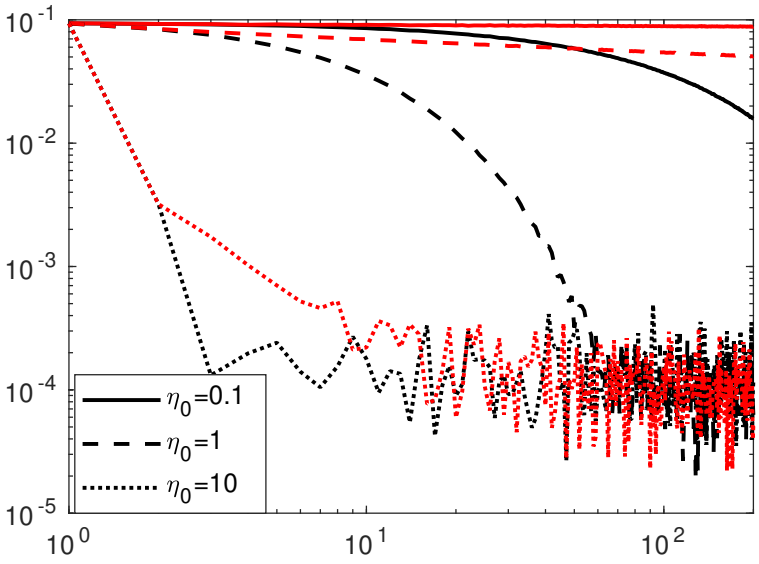


Figure 2.3: Convergence of the gradient norm $\|\nabla f(u_j)\|$ for different initial stepsizes, using either the SGD (red) or AdaGrad (black) method.

problem reduces the convergence rates of both algorithms, the AdaGrad method (in black) is more robust to the loss in convexity than the SGD method (in red), whose convergence breaks down for $\alpha = 0.01$.

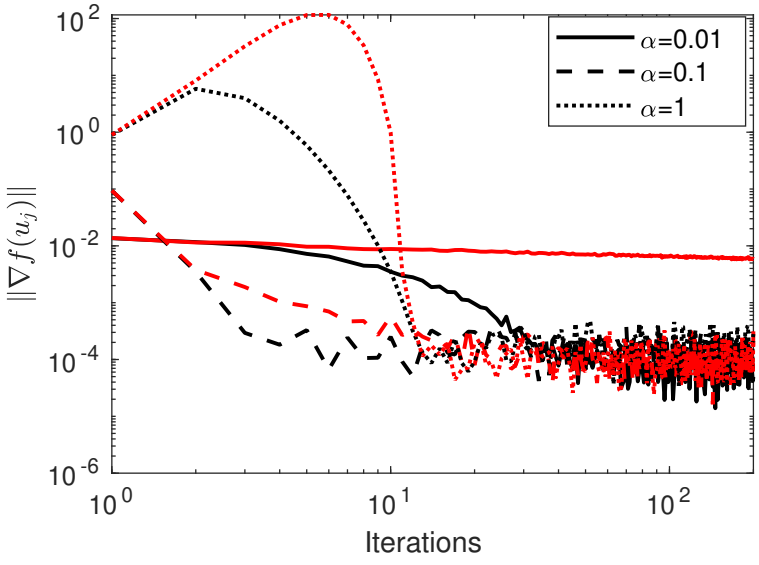


Figure 2.4: Convergence of the gradient norm $\|\nabla f(u_j)\|$ for different regularization parameters in the first 200 iterations of either the SGD (red) or the AdaGrad (black) method with an initial stepsize of $\eta_0 = 10$.

Example 2. In this example we consider a simple two-dimensional model of a lithium cell whose forcing term represents the heat generation rate associated with a power load resulting from two successive charge or discharge cycles.

The computational domain D, shown in Figure 2.5, represents a cross-section of a cylindrical cell with inner radius of 4mm, an outer radius of 32mm, and a length of 198mm. For simplicity, the surrounding temperature, as well as that in the axial center is assumed to be constant at $T_o = 18°\text{C}$.

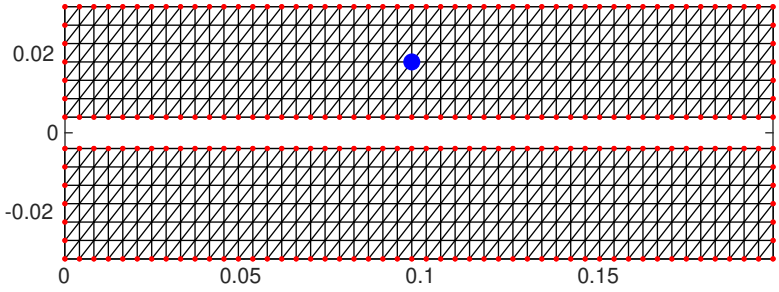


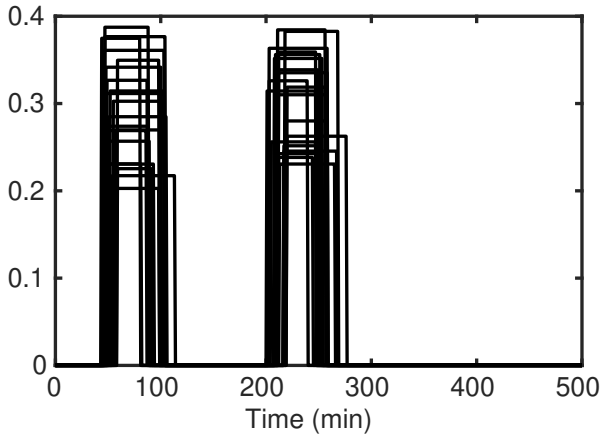
Figure 2.5: Diagram of the computational mesh for the two-dimensional model of the lithium cell.

The material is treated as a homogeneous solid with density $\rho = 2118$ kg m^{-3}, specific heat $c_p = 765$ J kg^{-1} K^{-1}, and anisotropic thermal conductivities in the horizontal and vertical directions given by $k_{x_1} = 66$ W m K^{-1} and $k_{x_2} = 0.66$ W m K^{-1} respectively (see [51]). The equation governing the temperature evolution inside the battery is therefore given by

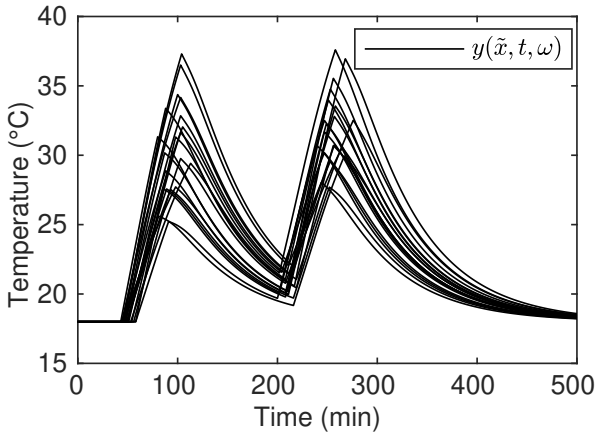
$$\begin{cases} \rho c_p \frac{\partial y}{\partial t} - k_{x_1} \frac{\partial^2 y}{\partial x_1^2} - k_{x_2} \frac{\partial^2 y}{\partial x_1^2} = g + u, & x \in D, t > 0 \\ y(x,0) = T_o, & x \in D \\ y(x,t) = T_o, & x \in \partial D, t > 0. \end{cases} \tag{2.31}$$

Heat is generated uniformly over the entire domain, but the timing, duration, and intensity of the power load are assumed to be independently and uniformly distributed uncertain quantities. Specifically, we take the onset time of the first charge pulse to range between 40 and 60 min and that of the second pulse between 200 and 220 min. The duration

of each pulse varies between 30 and 60 min, and their intensity lies between 200 and 400 W m^{-3}. Figure 2.6a shows 20 sample realizations of the resulting heat generation rate, while Figure 2.6b shows the associated uncontrolled temperature profiles at the fixed spatial location $\tilde{x} = (0.097, 0.098)$, indicated on Figure 2.5 by a blue dot.



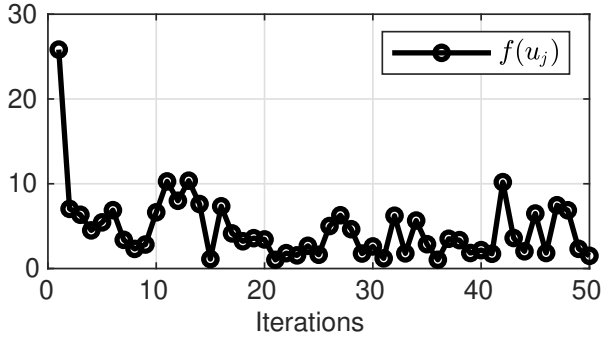
(a) Samples of the heat generation rate, scaled by ρ, c_p.



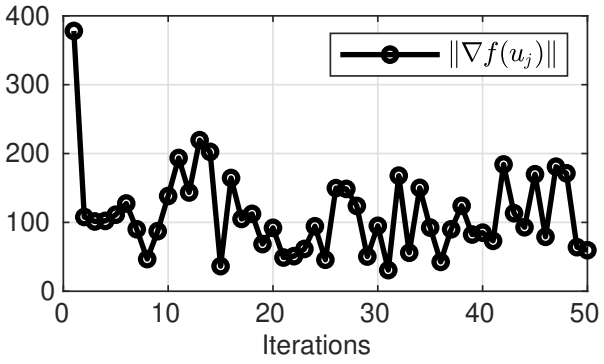
(b) Sample paths of the associated temperature with $u = 0$ at the point $\tilde{x}$.

Figure 2.6: Sample paths of the uncontrolled system.

We ran 50 iterations of the AdaGrad method with stepsize parameters $\eta = 0.1$ and $b_0 = 1$, an initial guess of $u_0 = 0$, and a target state of $y_d = 18°\text{C}$. Figures 2.7a and 2.7b show the convergence of the cost functional and of the gradient norm respectively. The persistence of significant random perturbations in the cost indicates far higher levels of variance in this system than were observed in Example 1.



(a) Convergence of the cost.



(b) Convergence of the gradient norm.

Figure 2.7: Convergence of the AdaGrad Method for Example 2 over 50 iterations.

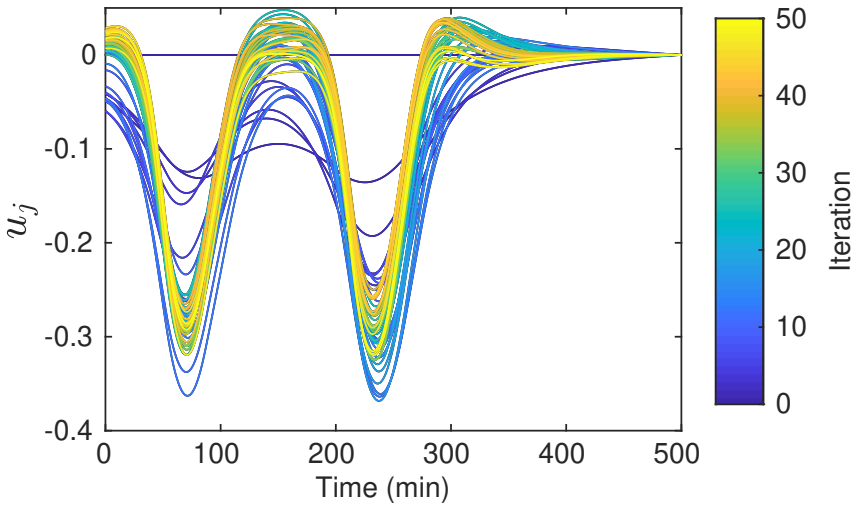


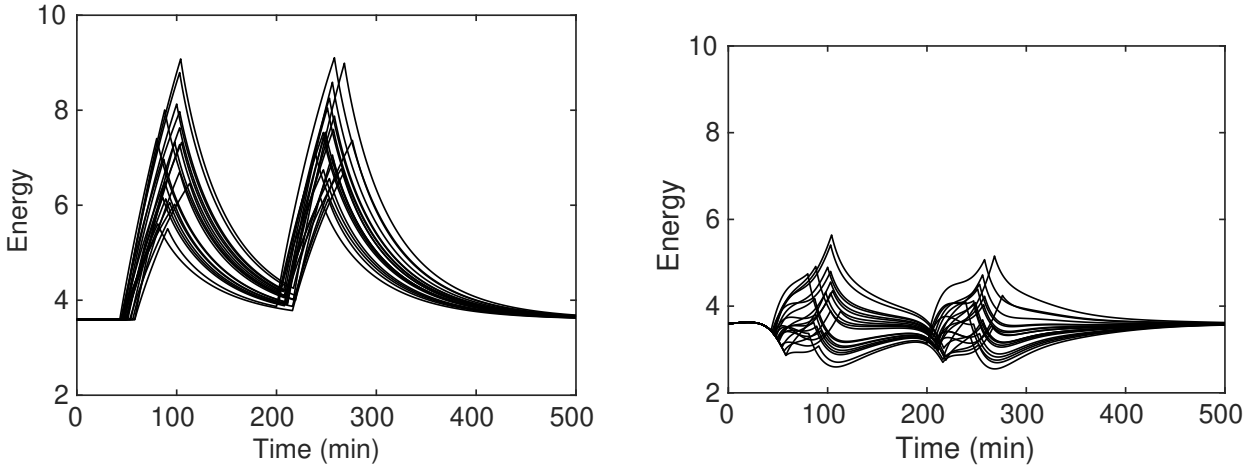
Figure 2.8: Iterates of the control function generated by the AdaGrad method, evaluated at the sample point $\tilde{x}$.

Figure 2.8 depicts the time evolution of the control iterates, evaluated at the sample point $\tilde{x}$, and shows how these converge to an optimal control counteracting the heat effect of the two uncertain pulses.

To show that the deterministic controller obtained by this stochastic optimization algorithm leads to a cooling of the system under a variety of conditions, we compare the time evolution of the system's heat energy $E(t) = \int_D y(x,t)^2 dx$ for the initial control (Figure 2.9a) and for the optimal control (Figure 2.9b). Evidently the addition of the optimal control leads to a significant reduction in heat energy over the entire time period.

2.4 Conclusion

This chapter shows the extension of the AdaGrad method to an infinite dimensional optimal control problem for the distributed control of a linear parabolic system. We related smoothness and finite variance requirements to the statistical distributions of the underlying model parameters and demonstrated how these can be used in thermal regulation of a simple model for an uncertain lithium battery system. It remains to explore how optimal control can be extended to more complex uncertain systems, how this approach can be incorporated



(a) Evolution of the heat energy with no control.

(b) Evolution of the heat energy with optimal control.

Figure 2.9: Comparison of the heat energy over time for the uncontrolled system (left) with that of the controlled one (right).

into existing battery management systems, and how existing battery degradation models and real-time observers can be used in designing such controls.

Chapter 3

Stochastic Alternating Least Squares method for Tensors

The work in this chapter appears in the paper *Analysis of the Stochastic Alternating Least Squares Method for the Decomposition of Random Tensors*, Y. Cao, S. Das, L. Oeding, and H.-W. van Wyk, 2021. [13]

3.1 Introduction

Multi-modal data is represented by a tensor, or a multidimensional matrix. Tensor data is present in areas such as Natural Language Processing (NLP) [10], Blind Source Separation (BSS) [11, 54], and Phylogenetic Tree Reconstruction (PTR) [18, 3, 29]. In each of these areas, canonical decomposition (CANDECOMP)/parallel factors (PARAFAC), also known as CP tensor decomposition (representing a given tensor as a sum of rank-one tensors), provides important insights since the components of the rank-one terms, the factors, represent meaning in the data. For NLP given a tensor constructed from large text databases, the rank-one terms could represent topics, and the factors could represent words. For BSS given a tensor constructed from signals arriving at a single receiver from unknown sources, the rank-one terms could represent sources, and the factors could be used to locate the sources. For PTR given a tensor constructed as a contingency table for instances of nucleotide combinations from aligned DNA from several species, the factors represent model parameters for the phylogenetic tree.

In many applications, tensor observations exhibit statistical variations that naturally suggest modeling them as samples from an underlying tensor-valued random variable, referred to herein as a *random tensor*. For example, the word co-occurence frequencies used in NLP may come from a sample of a collection of documents. Here fast, reliable algorithms are

desired for the CP decomposition of the tensor's *expected value*, obtained primarily through statistical sampling. Because CP decomposition is known to be NP hard in general [26] we investigate approximate decompositions based on stochastic optimization. These methods have also been used [57, 34] in conjunction with statistical subsampling for the decomposition of high dimensional deterministic tensors, expressible as expectations of their independent subsamples. In [39], Maehara et al. propose a stochastic version of the widely-used alternating least squares (ALS) method, the stochastic alternating least squares (SALS) method, and show the algorithm's efficiency by means of numerical experiments. A similar approach was discussed earlier in the context of the block-subsampling method proposed in [57]. In both these papers, the authors make convincing arguments for the use of stochastic optimization in tensor decomposition, both in terms storage and computational efficiency. In this work we provide a detailed analysis of the algorithm, showing under mild regularization and a minimal set of verifiable assumptions that it is convergent to a local stationary point for any initial guess. In 3.4, we also include a discussion of the algorithm's complexity and efficiency.

The alternating least squares (ALS) method, first proposed by Carroll and Chang [14], remains the most widely used algorithm for computing the CP decomposition of a tensor [20]. This block-nonlinear Gauss-Seidel method [8, 49] exploits the componentwise quadratic structure of the cost functional to compute iterates efficiently and with a low memory footprint. It has been modified [15] to exploit sparsity structure in data, which typically occurs in tensors representing co-occurence frequencies such as in the tree reconstruction above. Regularized versions of the ALS method, as well as the associated proximal minimization iterations considered in [36], help mitigate the potential ill-posedness of the CP problem to within sample noise. Although one may compute the tensor's expectation *a priori* and decompose it by means of the standard ALS method (see 5), such an approach results in a loss of sparsity and cannot seamlessly accommodate the arrival of new samples. In [39], Maehara

et al. proposed the Stochastic Alternating Least Squares method, a block-stochastic minimization method that preserves the salient features of the ALS method, while also efficiently incorporating tensor samples in the optimization procedure. Recent work (see [6, 34, 57]) has considered the related problem of using randomized (or stochastic) methods to decompose existing large-scale tensors. In particular, in [6], a variant of the ALS method is developed that approximates the component least squares problem efficiently by randomized/sketching methods. In [34], a dense tensor is expressed as the expectation of a sequence of sparse ones, thereby allowing the use of stochastic gradient descent (SGD) methods.

The convergence analysis of the SALS algorithm applied to the CP decomposition of a random tensor is complicated by the fact that the underlying cost functional is not convex. Moreover, the algorithm itself is a stochastic, block-iterative method, whose successive iterates do not have the Markovian dependence structure present in classical stochastic gradient descent (SGD) methods. The convergence of the ALS method was studied in various works (see [36, 58, 56] and references therein). Block-coordinate techniques for the unconstrained optimization of general, possibly nonconvex, deterministic functionals were treated in [24] (see also [47, 7, 63] and references therein). Xu and Yin [64] discuss the convergence of block stochastic *gradient* methods for averages of convex and nonconvex functionals. They rely on assumptions (such as the uniformly bounded variance of gradient iterates) that, while standard in the literature (see e.g. [9]), are difficult to verify in practice since they pertain to iterates of the algorithm itself. The objectives of this chapter are to prove the convergence of the SALS algorithm, a block-stochastic *Newton* method, for the CP decomposition of the average of a *random* tensor, subject to a single verifiable assumption relating to the boundedness of the observed data.

This chapter is structured as follows. In 3.2, we introduce the CP decomposition problem for random tensors and describe the stochastic alternating least squares algorithm (4). 3.3 contains our main theoretical contributions. In 3.3.2, we exploit the special multinomial structure of the cost functional to quantify the regularity of its componentwise gradient and

Hessian in terms of the size of the algorithm's iterate vectors (see 4 and its corollaries). In 3.3.1 we show that the iterates themselves can be bounded in terms of the random tensor to be decomposed (2), which naturally leads to our single, verifiable assumption on the latter's statistical properties (4). In 3.3.3, we show that the iterates generated by the SALS algorithm converge to a local minimizer. We validate the SALS method numerically via a few computational examples in 3.4 and offer concluding remarks in 3.5.

3.2 Notation and Problem Description

We follow notational conventions that are closely aligned with the literature on the CP decomposition (see [33]), as well as on stochastic optimization (see [9, 64]). We use lower case letters to denote scalars, bold letters for vectors, uppercase letters for matrices, and uppercase calligraphic letters for tensors. The (possibly subscripted) letters C and M are reserved for generic constants. We use superscripts to indicate iteration (or sub-iteration) indices and subscripts for components, i.e. x_i^k is the i-th component at the k-th iteration. Multi-indices are denoted by bold letters and sums, products, and integrals over these should be interpreted to be in iterated form. For the block coordinate minimization method in 4, it is convenient to express the vector $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_p)$ in terms of its i-th block component $\boldsymbol{x}_i$ and the remaining components, denoted $\boldsymbol{x}_{i^*} := (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_p)$. Using this notation, we write $\boldsymbol{x} = (\boldsymbol{x}_i, \boldsymbol{x}_{i^*})$ with the implicit understanding that the block components are in the correct order. The Frobenius norm of a vector, matrix, or tensor, i.e. the root mean square of its entries, is denoted by $\|\cdot\|$. For $1 \leq p \leq \infty$, $\|\cdot\|_p$ denotes the standard Euclidean p-norm for vectors and the induced matrix p-norm for matrices. We use '$\circ$' to denote the outer product, '$*$' for the componentwise (or Hadamard) product, '$\odot$' for the column-wise Khatri-Rao product, and '$\otimes$' for the Kronecker product.

Let $(\Omega, \mathcal{F}, d\mu)$ be a complete probability space and let $\mathcal{X} : \Omega \to \mathbb{R}^{n_1 \times n_2 \ldots \times n_p}$ be a measurable map, also known as a p-th order random tensor. In practice the underlying probability space is rarely known explicitly. Instead, its law can often be observed indirectly

through sample realizations $\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^n$ of $\mathcal{X}$ that are assumed to be independent and identically distributed (iid). We use $\mathbb{E}$ to denote expectation:

$$\mathbb{E}[f] = \int_{\Omega} f(\mathcal{X}) d\mu_{\mathcal{X}}$$

for any integrable function $f : \mathbb{R}^{n_1 \times \ldots \times n_p} \to \mathbb{R}^m$.

The rank-r decomposition problem for $\mathcal{X}$ (or its sample realizations) amounts to finding a set of r rank-one deterministic tensors $\{\hat{\mathcal{X}}_j\}_{j=1}^r$, so that the quantity

$$\hat{\mathcal{X}} := \sum_{j=1}^{r} \hat{\mathcal{X}}_j \tag{3.1}$$

is a good overall representation of $\mathcal{X}$. Each rank-one tensor $\hat{\mathcal{X}}_j$, $j = 1, \ldots, r$, is formed by the outer product $\hat{\mathcal{X}}_j = \boldsymbol{a}_{1,j} \circ \ldots \circ \boldsymbol{a}_{p,j}$ where $\boldsymbol{a}_{i,j} = (a_{i,j,1}, \ldots, a_{i,j,n_i}) \in \mathbb{R}^{n_i}$ for each $i = 1, \ldots p$, i.e. $\hat{\mathcal{X}}_j \in \mathbb{R}^{n_1 \times \ldots \times n_p}$ is defined componentwise by

$$\hat{\mathcal{X}}_{j,i_1,\ldots,i_p} = \prod_{l=1}^{p} a_{l,j,i_l}.$$

We use the mean squared error $\mathbb{E}\left[\|\mathcal{X} - \hat{\mathcal{X}}\|^2\right]$ to measure the quality of the representation. Other risk measures may be more appropriate, depending on the application, possibly requiring a different analysis and approximation technique.

For analysis and computation, it is convenient to consider two other representations of the design variable. We define the i-th factor matrix $A_i = [\boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,r}]$ in $\mathbb{R}^{n_i \times r}$ to consist of the i-th component vectors of each term in decomposition (3.1). Let $\boldsymbol{x} = \mathrm{vec}([A_1, \ldots, A_p]) \in \mathbb{R}^{nr}$, with $n = \sum_{i=1}^{p} n_i$, be the vectorization of the factor matrices, i.e. the concatenation of their column vectors. We also write $\boldsymbol{x}$ in block component form as $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$, where $\boldsymbol{x}_i = \mathrm{vec}(A_i) \in \mathbb{R}^{rn_i}$ for $i = 1, \ldots, p$. To emphasize dependence of $\hat{\mathcal{X}}$ on $\boldsymbol{x}$, we write the

following (which also defines $[\![\cdot]\!]$)

$$\hat{\mathcal{X}} = [\![\boldsymbol{x}]\!] = [\![A_1, \ldots, A_p]\!] := \sum_{j=1}^{r} \boldsymbol{a}_{1,j} \circ \ldots \circ \boldsymbol{a}_{p,j}.$$

No *efficient* closed form solution of the factorization problem exists (even for deterministic tensors) [26]. So it is commonly reformulated as the optimization problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^{nr}} \mathbb{E}\left[\| \mathcal{X} - [\![\boldsymbol{x}]\!] \|^2 \right]. \tag{3.2}$$

Remark 5. *Letting* $\boldsymbol{i} = (i_1, \ldots, i_p)$ *and* $\boldsymbol{n} = (n_1, \ldots, n_p)$*, it follows readily that*

$$\begin{aligned}
\mathbb{E}\left[\| \mathcal{X} - [\![\boldsymbol{x}]\!] \|^2 \right] &= \mathbb{E}\left[\sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \left(\mathcal{X}_{\boldsymbol{i}} - [\![\boldsymbol{x}]\!]_{\boldsymbol{i}} \right)^2 \right] \\
&= \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \mathbb{E}\left[\mathcal{X}_{\boldsymbol{i}}^2 \right] - 2\mathbb{E}\left[\mathcal{X}_{\boldsymbol{i}} \right] [\![\boldsymbol{x}]\!]_{\boldsymbol{i}} + [\![\boldsymbol{x}]\!]_{\boldsymbol{i}}^2 \\
&= \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \operatorname{var}(\mathcal{X}_{\boldsymbol{i}}) + \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \left(\mathbb{E}(\mathcal{X}_{\boldsymbol{i}}) - [\![\boldsymbol{x}]\!]_{\boldsymbol{i}} \right)^2.
\end{aligned}$$

Since the variance $\operatorname{var}(\mathcal{X}_{\boldsymbol{i}})$ *of* $\mathcal{X}_{\boldsymbol{i}}$ *is constant in the design variable* $\boldsymbol{x}$*, the minimization Problem 3.2 is equivalent to the decomposition of the expected tensor, i.e.*

$$\min_{x \in \mathbb{R}^{nr}} \frac{1}{2} \| \mathbb{E}[\mathcal{X}] - [\![\boldsymbol{x}]\!] \|^2.$$

This formulation suggests that the cost function's partial Hessian is independent of $\mathcal{X}$ *and hence deterministic, a result we confirm in* (3.8) *and will use repeatedly in our analysis.*

Tensor decompositions have scale ambiguity. Indeed, for any direction $i = 1, .., p$ and any scalars $\beta_{i,1}, \ldots, \beta_{i,r-1} > 0$, let $\beta_{i,r} = 1/\prod_{j=1}^{r-1} \beta_{i,j}$. Then

$$\sum_{j=1}^{r} (\beta_{1,j} \boldsymbol{a}_{1,j}) \circ \ldots \circ (\beta_{p,j} \boldsymbol{a}_{p,j}) = \sum_{j=1}^{r} \boldsymbol{a}_{1,j} \circ \ldots \circ \boldsymbol{a}_{p,j}.$$

The optimizer of (3.2) therefore lies on a simply connected manifold (see [56]), which can lead to difficulties in the convergence of optimization algorithms. To ensure isolated minima that can be readily located, it is common to enforce bounds on the size of the factors, either directly through an appropriate normalization (see [56]), or indirectly through the addition of a regularization term (see e.g. [45, 36]). We pursue the latter strategy, leading to the problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^{nr}} F(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^{nr}} \mathbb{E}\left[f(\boldsymbol{x})\right], \text{ with}$$
$$f(\boldsymbol{x};\mathcal{X}) = \frac{1}{2}\left\|\mathcal{X} - [\![\boldsymbol{x}]\!]\right\|^2 + \frac{\lambda}{2}\|\boldsymbol{x}\|^2, \qquad \boldsymbol{x}\in\mathbb{R}^{nr}, \lambda > 0. \quad (3.3)$$

While the regularization term biases the minimizers of Problem (3.2) its inclusion is key to guaranteeing well-posedness in the presence of noise. It plays a pivotal role in (i) proving the existence of a minimizer (1), (ii) ensuring that the partial Hessians with respect to each block variable remain positive definite (5), and (iii) guaranteeing the boundedness of iterates in terms of random tensor $\mathcal{X}$ (2). Heuristic methods are typically used to choose the value of λ that balances the bias of the optimizers against stability considerations, the most well-known of which is the Morozov discrepancy principle [44]. The cost function in (3.3) is continuous. Moreover, the regularization term ensures that it has bounded sublevel sets. The existence of a minimizer, as stated in the following lemma, is an immediate consequence thereof.

Lemma 1 (Existence of Minimizers)**.** *Problem 3.3 has at least one minimizer.*

Proof. Let $F^* = \inf_{\boldsymbol{x}\in\mathbb{R}^{nr}} F(\boldsymbol{x})$. So there exists a sequence $\{\boldsymbol{x}^k\}_{k=1}^{\infty}$ in $\mathbb{R}^{nr}$ with

$$lim_{k\to\infty} F(\boldsymbol{x}^k) = F^*,$$

from which it follows that $F(\boldsymbol{x}^k)$ is bounded. The inequality

$$\|\boldsymbol{x}\|^2 \leq \frac{2}{\lambda} F(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^{rn}, \tag{3.4}$$

allows us to establish the boundedness of $\{\boldsymbol{x}^k\}_{k=1}^{\infty}$. By compactness, there exists a convergent subsequence $\boldsymbol{x}^{k_i} \to \boldsymbol{x}^*$ as $i \to \infty$. The continuity of F then implies

$$F(\boldsymbol{x}^*) = \lim_{i \to \infty} F(\boldsymbol{x}^{k_i}) = F^*.$$

□

Remark 6. *For the sample realizations $f(\boldsymbol{x}; \mathcal{X})$ we have a bound similar to* (3.4)*:*

$$\textit{for all } \boldsymbol{x} \in \mathbb{R}^{nr}, \quad \|\boldsymbol{x}\|^2 \leq \frac{2}{\lambda} f(\boldsymbol{x}; \mathcal{X}), \quad \textit{a.s. on } \Omega. \tag{3.5}$$

3.2.1 The Deterministic Alternating Least Squares Method

Although the stochastic integrand $f(\boldsymbol{x}; \mathcal{X})$, and hence the cost functional $F(\boldsymbol{x})$, is a high degree polynomial in general, it is quadratic in each component factor matrix A_i. This is apparent from the matricized form of (3.3). Recall [31] that the columnwise Khatri-Rao product $\odot : \mathbb{R}^{n_i \times r} \times \mathbb{R}^{n_j \times r} \to \mathbb{R}^{n_i n_j \times r}$ of matrices $A = [\boldsymbol{a}_1, ..., \boldsymbol{a}_r]$ and $B = [\boldsymbol{b}_1, ..., \boldsymbol{b}_r]$ is defined as their columnwise Kronecker product, i.e. $A \odot B = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1, ..., \boldsymbol{a}_r \otimes \boldsymbol{b}_r]$. The matricization $[\![A_1, \ldots, A_p]\!]_{(i)}$ of the rank r tensor $[\![A_1, \ldots, A_p]\!]$ along the i-th fiber takes the form (see [1])

$$[\![A_1, \ldots, A_p]\!]_{(i)} = A_i \left(A_p \odot \ldots \odot A_{i+1} \odot A_{i-1} \odot \ldots \odot A_1\right)^T =: A_i \Theta_i^T, \tag{3.6}$$

where Θ_i is defined to be the iterated Khatri-Rao product on the right. Note that Θ_i does not depend on A_i, and hence the matricized tensor decomposition is linear in A_i. Since the Frobenius norm is invariant under matricization, the sample objective function $f(\boldsymbol{x}; \mathcal{X})$ can

then be rewritten as a quadratic function in A_i, i.e.

$$f(A_1, \ldots, A_p; \mathcal{X}) := \frac{1}{2} \left\| \mathcal{X}_{(i)} - A_i \Theta_i^T \right\|^2 + \frac{\lambda}{2} \sum_{j=1}^{p} \|A_j\|^2.$$

Vectorizing this form yields a linear least squares objective function in $\boldsymbol{x}_i$, namely

$$f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p; \mathcal{X}) = \frac{1}{2} \left\| \mathrm{vec}(\mathcal{X}_{(i)}) - (\Theta_i \otimes I_{n_i}) \boldsymbol{x}_i \right\|^2 + \frac{\lambda}{2} \sum_{j=1}^{p} \|\boldsymbol{x}_j\|^2,$$

whose componentwise gradient and Hessian are given respectively by

$$\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) = -(\Theta_i^T \otimes I_{n_i}) \mathrm{vec}(\mathcal{X}_{(i)}) + \left((\Theta_i^T \Theta_i + \lambda I_r) \otimes I_{n_i} \right) \boldsymbol{x}_i, \quad \text{and} \tag{3.7}$$

$$\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}) = (\Theta_i^T \Theta_i + \lambda I_r) \otimes I_{n_i}, \tag{3.8}$$

where $I_r \in \mathbb{R}^{r \times r}$ and $I_{n_i} \in \mathbb{R}^{n_i \times n_i}$ are identity matrices. In the presence of regularization, the partial Hessian $\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})$ is strictly positive definite with lower bound that is independent of both $\mathcal{X}$ and of the remaining components $\boldsymbol{x}_{i^*}$ (see 5). Consequently, each sampled componentwise problem

$$\min_{\boldsymbol{x}_i \in \mathbb{R}^{r n_i}} f(\boldsymbol{x}; \mathcal{X})$$

has a unique solution given by the stationary point $\boldsymbol{x}_i$ satisfying $\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}, \mathcal{X}) = 0$. It is more efficient to use the matricized form of the stationarity condition, i.e.

$$0 = -\mathcal{X}_{(i)} \Theta_i + A_i (\Theta_i^T \Theta_i + \lambda I_r), \tag{3.9}$$

yielding the stationary point

$$A_i = \mathcal{X}_{(i)} \Theta_i (\Theta_i^T \Theta_i + \lambda I_r)^{-1}. \tag{3.10}$$

For any $A \in \mathbb{R}^{n_i \times r}, B \in \mathbb{R}^{n_j \times r}$, it can be shown [55] that

$$(A \odot B)^T (A \odot B) = (A^T A) * (B^T B),$$

where $*$ denotes the Hadamard product. Repeatedly using this identity, we have

$$\Theta_i^T \Theta_i := (A_1^T A_1) * \ldots * (A_{i-1}^T A_{i-1}) * (A_{i+1}^T A_{i+1}) * \ldots * (A_p^T A_p), \tag{3.11}$$

so the partial Hessian can be computed as the entry-wise product of p matrices in $\mathbb{R}^{r \times r}$. We relate the sample componentwise minimizer satisfying (3.9) to that of $F(\boldsymbol{x})$ by noting that according to (3.8) the partial Hessian matrix $\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})$ is independent of $\mathcal{X}$ and that $\nabla^2_{\boldsymbol{x}_i} F(\boldsymbol{x}) = \nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})$ is positive definite. Moreover, the partial gradient $\nabla f_{\boldsymbol{x}_i}(\boldsymbol{x}; \mathcal{X})$ in (3.7) is linear in $\mathcal{X}$ and hence $\nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}) = \mathbb{E}\left[\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})\right] = \nabla_{\boldsymbol{x}_i} f\left(\boldsymbol{x}; \mathbb{E}[\mathcal{X}]\right)$. The componentwise minimizer of $F(\boldsymbol{x})$ therefore takes the form

$$A_i = \mathbb{E}[\mathcal{X}_{(i)}] \Theta_i (\Theta_i^T \Theta_i + \lambda I)^{-1}, \tag{3.12}$$

which resembles Equation (3.9) with $\mathcal{X}_{(i)}$ replaced by $\mathbb{E}[\mathcal{X}_{(i)}]$. The (deterministic) ALS method (algorithm 3) exploits this componentwise quadratic structure of the objective functional F. In the k-th block iteration, the ALS algorithm cycles through the p components $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p$ of $\boldsymbol{x}$, updating each component in turn in the direction of the componentwise minimizer. The function is also updated at each subiteration to reflect this change. Specifically, the iterate at the beginning of the k-th block is denoted by

$$\boldsymbol{x}^k = \boldsymbol{x}^{k,0} = (\boldsymbol{x}_1^k, \boldsymbol{x}_2^k, ..., \boldsymbol{x}_p^k).$$

The ALS algorithm then generates a sequence of sub-iterates $\boldsymbol{x}^{k,1}, ..., \boldsymbol{x}^{k,p}$, where

$$\boldsymbol{x}^{k,i} = (\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{x}_i^{k+1}, \boldsymbol{x}_{i+1}^k, \ldots, \boldsymbol{x}_p^k).$$

Note that, under this convention, $\boldsymbol{x}^{k,p} = \boldsymbol{x}^{k+1} = \boldsymbol{x}^{k+1,0}$.

Algorithm 3 The Alternating Least Squares Algorithm

1: Initial guess $\boldsymbol{x}^1$
2: **for** $k = 1, 2, \ldots$ **do**
3: **for** $i = 1, \ldots, p$ **do**
4: $\boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i \in \mathbb{R}^{rn_i}}{\operatorname{argmin}} F(\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_p^{k})$
5: $\boldsymbol{x}^{k,i} = (\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_i^{k+1}, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_p^{k})$
6: **end for**
7: **end for**

Although the descent achieved by the ALS method during k-th block iteration is most likely not as large as a descent would be for a monolithic descent direction for F over the entire space $\mathbb{R}^{nr}$, the ALS updates can be obtained at a significantly lower computational cost and with a lower memory footprint.

In anticipation of the stochastic-optimization-based SALS algorithm introduced in the following section, we express the componentwise update $\boldsymbol{x}_i^k \to \boldsymbol{x}_i^{k+1}$ in terms of a descent step. To this end we form the second order componentwise Taylor expansion of F about $\boldsymbol{x}_i$ at the current iterate $\boldsymbol{x}^{k,i-1}$, i.e.

$$F(\boldsymbol{x}_1^{k+1}, \ldots \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{x}_i + \boldsymbol{p}, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_p^{k}) = F^{k,i} + (\boldsymbol{g}^{k,i})^T \boldsymbol{p} + \frac{1}{2} \boldsymbol{p}^T H^{k,i} \boldsymbol{p}, \tag{3.13}$$

$$\begin{aligned} \text{with} \quad & F^{k,i} = F(\boldsymbol{x}^{k,i-1}) = \mathbb{E}\left[f(\boldsymbol{x}^{k,i-1}) \right], \\ & \boldsymbol{g}^{k,i} = \nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i-1}) = \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}; \mathbb{E}\left[\mathcal{X}\right]), \quad \text{and} \\ & H^{k,i} = \nabla^2_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i-1}) = (\Theta_i^T \Theta_i + \lambda I_r) \otimes I_{n_i} \end{aligned}$$

The componentwise minimizer $\boldsymbol{x}_i^{k+1}$ of F can be calculated explicitly as the Newton update

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k - (H^{k,i})^{-1} \boldsymbol{g}^{k,i}.$$

3.2.2 The Stochastic Alternating Least Squares Method

The ALS method described above approximates the expectation $\mathbb{E}[\mathcal{X}]$ and uses it to generate the componentwise minimizers A_i via Formula 3.12. For a high dimensional tensor $\mathcal{X}$ with a dense expectation $\mathbb{E}[\mathcal{X}]$ the associated computational cost can be considerable. Indeed, recall that r is the specified rank, p the tensor's dimension, and $n = \sum_{i=1}^{p} n_i$ where n_i is the size of the i-th vector component of each outer product in the expansion. Further, let $\text{nnz}(\mathbb{E}[\mathcal{X}])$ be the number of non-zero entries in $\mathbb{E}[\mathcal{X}]$. It can then readily be seen that the cost of forming the coefficient matrix $\Theta_i^T\Theta_i + \lambda I_r$ is $O(pr^2n)$, that of computing the matricized tensor times Khatri-Rao product (MTTKRP) $\mathbb{E}[\mathcal{X}_{(i)}]\Theta_i$ is $O(pr \cdot \text{nnz}(\mathbb{E}[\mathcal{X}_{(i)}]))$, and that of solving the resulting dense symmetric system is $O(pr^3)$. If $r \ll \text{nnz}(\mathbb{E}[\mathcal{X}])$, the computational cost of the MTTKRP dominates, especially as $\text{nnz}(\mathbb{E}[\mathcal{X}_{(i)}]) \to \prod_{i=1}^{p} n_i$, the total number of entries in $\mathbb{E}[\mathcal{X}]$.

In cases where the sample realizations of $\mathcal{X}$ are sparse despite the density of $\mathbb{E}[\mathcal{X}]$, the computation of sample componentwise minimizers by means of Equation 3.10 is considerably cheaper. This suggests the use of stochastic gradient sampling algorithms that efficiently incorporate the sampling procedure into the optimization iteration, thereby exploiting the inherent sparsity in the data. The stochastic gradient descent method [52] addresses the cost of approximating the expectation by computing descent directions from small sampled batches of function values and gradients at each step of the iteration. To accommodate the noisy gradients, the stepsize is reduced at a predetermined rate. For the stochastic alternating least squares (SALS) method (4), we determine the sample at the beginning of the k-th block iteration. For a batch $\boldsymbol{\mathcal{X}} = (\mathcal{X}^1, \ldots, \mathcal{X}^m)$ of m iid random samples of $\mathcal{X}$, we define the batch averages $\tilde{\mathcal{X}} = \frac{1}{m}\sum_{l=1}^{m} \mathcal{X}^l$ and $\tilde{f}(\boldsymbol{x}; \boldsymbol{\mathcal{X}}) = \frac{1}{m}\sum_{l=1}^{m} f(\boldsymbol{x}; \mathcal{X}^l)$. As before, we can compute the componentwise minimizer

$$\hat{\boldsymbol{x}}_i^{k+1} = \underset{\boldsymbol{x}_i \in \mathbb{R}^{n_i}}{\operatorname{argmin}} \tilde{f}(\boldsymbol{x}_1^{k+1}, \ldots \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^{k}, \ldots, \boldsymbol{x}_p^{k}; \boldsymbol{\mathcal{X}}) \tag{3.14}$$

at the i-th subiteration by means of the Newton step. To this end we express $\tilde{f}$ in terms of its second order Taylor expansion about the current iterate $\boldsymbol{x}^{k,i-1}$, i.e.

$$\tilde{f}(\boldsymbol{x}_1^{k+1},\ldots,\boldsymbol{x}_{i-1}^{k+1},\boldsymbol{x}_i+\boldsymbol{p},\boldsymbol{x}_{i+1}^{k},\ldots,\boldsymbol{x}_p^{k};\boldsymbol{\mathcal{X}}) = \tilde{f}^{k,i} + (\tilde{\boldsymbol{g}}^{k,i})^T\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T H^{k,i}\boldsymbol{p},$$

with

$$\begin{aligned}\tilde{f}^{k,i}(\boldsymbol{\mathcal{X}}) &= \frac{1}{m}\sum_{l=1}^{m} f(\boldsymbol{x}^{k,i-1};\mathcal{X}^l),\\ \tilde{\boldsymbol{g}}^{k,i}(\boldsymbol{\mathcal{X}}) &= \frac{1}{m}\sum_{l=1}^{m} \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1};\mathcal{X}^l) = \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1};\tilde{\mathcal{X}}^k),\end{aligned}$$

and $H^{k,i}$ as defined in 3.13. The sample minimizer is thus

$$\hat{\boldsymbol{x}}_i^{k+1} = \boldsymbol{x}_i^k - \left(H^{k,i}\right)^{-1}\tilde{\boldsymbol{g}}^{k,i}.$$

To mitigate the effects of noise on the estimate, especially during later iterations, we modify the update by introducing a variable stepsize parameter $\alpha^{k,i} > 0$, so that

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k - \alpha^{k,i}\left(H^{k,i}\right)^{-1}\tilde{\boldsymbol{g}}^{k,i}. \tag{3.15}$$

It is well known (see e.g. [9]) that a stepsize $\alpha^{k,i}$ that decreases at the rate of $O\left(\frac{1}{k}\right)$ as $k\to\infty$ leads to an optimal convergence rate for stochastic gradient methods. Here, we specify that the stepsize takes the form

$$\alpha^{k,i} = \frac{c^{k,i}}{k}, \qquad \text{for } i = 1,2,\ldots,p, \text{ and } k = 1,2,\ldots, \tag{3.16}$$

where $c^{k,i}$ is required to be uniformly positive and bounded above, i.e. there are constants $c_{\min}, c_{\max} \in (0, 2]$ so that

$$0 < c_{\min} \leq c^{k,i} \leq c_{\max} \leq 2, \qquad \text{for all } k = 1, 2, \ldots, \ i = 1, 2, \ldots, p.$$

The SALS algorithm is outlined in 4 below.

Algorithm 4 Stochastic Alternating Least Squares Algorithm

1: Initial guess $\boldsymbol{x}^1$
2: **for** $k = 1, 2, \ldots$ **do**
3: Generate a random sample $\boldsymbol{\mathcal{X}}^k = [\mathcal{X}^{k,1}, \ldots, \mathcal{X}^{k,m_k}]$
4: **for** $i = 1, \ldots, p$ **do**
5: Compute sample gradient $\tilde{\boldsymbol{g}}^{k,i}$ and Hessian $H^{k,i}$
6: Compute stepsize $\alpha^{k,i} = \frac{c^{k,i}}{k}$
7: Update i-th block $\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k - \alpha^{k,i} \left(H^{k,i}\right)^{-1} \tilde{\boldsymbol{g}}^{k,i}$ so that $\boldsymbol{x}^{k,i} = (\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_i^{k+1}, \boldsymbol{x}_{i+1}^k, \ldots, \boldsymbol{x}_p^k)$
8: **end for**
9: $\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k,p}$
10: **end for**

In practice, Step 7 in 4 is computed similarly to the update in 3. Specifically, $\hat{\boldsymbol{x}}_i^{k+1}$ can be written in matricized form as

$$\hat{A}_i^{k+1} = \tilde{\mathcal{X}}_{(i)}^k \Theta_i (\Theta_i^T \Theta_i + \lambda I)^{-1},$$

where $\tilde{\mathcal{X}}^k = \frac{1}{m_k} \sum_{l=1}^{m_k} \mathcal{X}^{k,l}$ is the k-th batch average. Using the update rule 3.15, the matricized component update A_i^{k+1} can be computed in terms of $\hat{A}_i^{k+1}$ as

$$A_i^{k+1} = \alpha^{k,i} \hat{A}_i^{k+1} + (1 - \alpha^{k,i}) A_i^k.$$

One of the difficulties in analyzing the convergence behavior of stochastic sampling methods arises from the fact that the iterates $\boldsymbol{x}^{k,i}$ constitute a stochastic process generated by a stochastic algorithm that depends on the realizations of the sample batches $\boldsymbol{\mathcal{X}}^1, \boldsymbol{\mathcal{X}}^2, \ldots \boldsymbol{\mathcal{X}}^{k-1}$. Consequently, even deterministic functions, such as F, become stochastic

when evaluated at these points, e.g. $F(\boldsymbol{x}^{k,i})$ is a random quantity. Moreover, successive iterates are statistically dependent, since later iterates are updates of earlier ones. Specifically, let $\mathcal{F}^k = \sigma(\boldsymbol{\mathcal{X}}^1, \ldots, \boldsymbol{\mathcal{X}}^k)$ be the σ-algebra generated by the first k sample batches , i.e. $\mathcal{F}^k = \{(\boldsymbol{\mathcal{X}}^1)^{-1}(A) \times \ldots \times (\boldsymbol{\mathcal{X}}^k)^{-1}(A) : A \in \mathcal{B}((\mathbb{R}^{n_1 \times \ldots \times n_p})^k)\}$, where $\mathcal{B}$ denotes the collection of Borel sets. Simply put, $\mathcal{F}^k$ represents the information contained in the first k tensor sample batches. At the end of the i-th subiteration in the k-th block, the first i components of $\boldsymbol{x}^{k,i} = (\boldsymbol{x}_1^{k+1}, ..., \boldsymbol{x}_i^{k+1}, \boldsymbol{x}_{i+1}^k, ..., \boldsymbol{x}_p^k)$ have been updated using $\boldsymbol{\mathcal{X}}^k$ and are thus $\mathcal{F}^k$-measurable, whereas the remaining components $\boldsymbol{x}_{i+1}^k, ..., \boldsymbol{x}_p^k$ depend only on $\boldsymbol{\mathcal{X}}^1, ..., \boldsymbol{\mathcal{X}}^{k-1}$ and are therefore only $\mathcal{F}^{k-1}$-measurable. To separate the effect of $\boldsymbol{\mathcal{X}}^k$ from that of $\boldsymbol{\mathcal{X}}^1, \ldots, \boldsymbol{\mathcal{X}}^{k-1}$ on an $\mathcal{F}^k$-measurable random variable h, it is often useful to invoke the law of total expectation, i.e.

$$\mathbb{E}_{\boldsymbol{\mathcal{X}}^1, ..., \boldsymbol{\mathcal{X}}^{k-1}, \boldsymbol{\mathcal{X}}^k}[h] = \mathbb{E}_{\boldsymbol{\mathcal{X}}^1, ..., \boldsymbol{\mathcal{X}}^{k-1}} \left[\mathbb{E}_{\boldsymbol{\mathcal{X}}^k} \left[h | \mathcal{F}^{k-1} \right] \right].$$

3.3 Convergence Analysis

Here we discuss the convergence of Algorithm 4. Since Problem (3.3) is nonconvex, there can in general be no unique global minimizer. 3.1 establishes the mean squared convergence of the SALS iterates $\boldsymbol{x}^{k,i}$ to a set of stationary points. Yet, the special structure of the CP problem allows us to forego most of the standard assumptions made in the SGD framework. In fact, we only assume boundedness of the sampled data $\mathcal{X}$. Indeed, we show in 3.3.1 that the norm of the iterates $\boldsymbol{x}^{k,i}$ are bounded by $\|\mathcal{X}\|$ in (2). Since the regularity estimates derived in 3.3.2 all involve powers of $\|\boldsymbol{x}\|$ and of $\|\boldsymbol{\mathcal{X}}^k\|$, this suggests that a bound on the data $\mathcal{X}$ is sufficient to guarantee the regularity of the cost functional, gradient, and componentwise Newton steps that are necessary to show convergence.

3.3.1 Boundedness of the Data

In this section we bound the norm of the component iterates $\boldsymbol{x}_i^k$ in terms of the norm of the initial guess and maximum value of the sequence of sample averages of the norms of $\mathcal{X}$.

Lemma 2. *The iterates $\boldsymbol{x}^{k,i}$ generated by Algorithm 4 satisfy*

$$\|\boldsymbol{x}_i^{k+1}\| \leq \max\left\{\|\boldsymbol{x}_i^1\|, R(\boldsymbol{\mathcal{X}}^k), \ldots, R(\boldsymbol{\mathcal{X}}^1)\right\}$$

for each $k = 1, 2 \ldots$, and $i = 1, \ldots, p$, where $R(\boldsymbol{\mathcal{X}}^k)$ is given by (3.17).

Proof. Note that the componentwise minimizer $\hat{\boldsymbol{x}}_i^{k+1}$ given in (3.14) satisfies

$$\begin{aligned}
&\|\hat{\boldsymbol{x}}_i^{k+1}\|^2 + \sum_{j=1}^{i-1} \|\boldsymbol{x}_j^{k+1}\|^2 + \sum_{j=i+1}^{p} \|\boldsymbol{x}_j^k\|^2 \\
&\leq \frac{2}{\lambda}\tilde{f}(\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_{i-1}^{k+1}, \hat{\boldsymbol{x}}_i^{k+1}, \boldsymbol{x}_{i+1}^k, \ldots, \boldsymbol{x}_p^k; \boldsymbol{\mathcal{X}}^k) && \text{(by 3.3)} \\
&\leq \frac{2}{\lambda}\tilde{f}(\boldsymbol{x}_1^{k+1}, \ldots, \boldsymbol{x}_{i-1}^{k+1}, \boldsymbol{0}, \boldsymbol{x}_{i+1}^k, \ldots, \boldsymbol{x}_p^k; \boldsymbol{\mathcal{X}}^k) && \text{(by optimality)} \\
&= \frac{1}{\lambda}\left(\frac{1}{m_k}\sum_{l=1}^{m_k} \|\mathcal{X}^{k,l}\|^2\right) + \sum_{j=1}^{i-1} \|\boldsymbol{x}_j^{k+1}\|^2 + \sum_{j=i+1}^{p} \|\boldsymbol{x}_j^k\|^2,
\end{aligned}$$

so that

$$\|\hat{\boldsymbol{x}}_i^{k+1}\| \leq \sqrt{\frac{1}{\lambda}\left(\frac{1}{m_k}\sum_{l=1}^{m_k} \|\mathcal{X}^{k,l}\|^2\right)} =: R(\boldsymbol{\mathcal{X}}^k). \tag{3.17}$$

To bound the iterates $\boldsymbol{x}_i^{k+1}$ note that Equation (3.15) can be rewritten as the convex combination

$$\boldsymbol{x}_i^{k+1} = \alpha^{k,i}\hat{\boldsymbol{x}}_i^{k+1} + (1 - \alpha^{k,i})\boldsymbol{x}_i^k,$$

which, by virtue of the stepsize bounds $0 \leq \alpha^{k,i} \leq 2$, implies

$$\begin{aligned}
\|\boldsymbol{x}_i^{k+1}\| &\leq |\alpha^{k,i}|\|\hat{\boldsymbol{x}}_i^{k+1}\| + |1 - a^{k,i}|\|\boldsymbol{x}_i^k\| \leq \max\{\|\boldsymbol{x}_i^k\|, R(\boldsymbol{\mathcal{X}}^k)\}(|\alpha^{k,i}| + |1 - \alpha^{k,i}|) \\
&\leq \max\{\|\boldsymbol{x}_i^k\|, R(\boldsymbol{\mathcal{X}}^k)\} \leq \max\left\{\|\boldsymbol{x}_i^{k-1}\|, R(\boldsymbol{\mathcal{X}}^k), R(\boldsymbol{\mathcal{X}}^{k-1})\right\} \leq \ldots \\
&\leq \max\left\{\|\boldsymbol{x}_i^1\|, R(\boldsymbol{\mathcal{X}}^k), R(\boldsymbol{\mathcal{X}}^{k-1}), \ldots, R(\boldsymbol{\mathcal{X}}^1)\right\}.
\end{aligned}$$

□

In light of 2, the following assumption guarantees the uniform boundedness of the iterates $\boldsymbol{x}^{k,i}$.

Assumption 4 (Bounded data)**.** *There is a constant* $0 < M < \infty$ *so that*

$$\|\mathcal{X}\| \leq M, \quad a.s.\ on\ \Omega. \tag{3.18}$$

Remark 7. *This assumption might conceivably be weakened to one pertaining to the statistical distribution of the maxima* $R(\boldsymbol{\mathcal{X}}^1), R(\boldsymbol{\mathcal{X}}^2), \ldots, R(\boldsymbol{\mathcal{X}}^k)$*. Specifically, letting* $r_k = \max_{l=1,\ldots,k} R(\boldsymbol{\mathcal{X}}^l)$*, it can be shown under appropriate conditions on the density of* $R(\mathcal{X}^k)$*, that* r_k *converges in distribution to a random variable with known extreme value density. The analysis below will hold if it can be guaranteed that the limiting distribution has bounded moments of sufficiently high order. This possibility will be pursued in future work.*

An immediate consequence of Assumption 4 is the existence of a uniform bound on the radius $R(\boldsymbol{\mathcal{X}})$ and hence on the iterates $\boldsymbol{x}^{k,i}$.

Corollary 1. *Given Assumption 4, there exist finite, non-negative constants* M_1, M_2*, and* M_3 *independent of* $\boldsymbol{x}^k$ *and of* $\mathcal{X}^k$*, so that for all* $k = 1, 2, \ldots,$

$$R(\boldsymbol{\mathcal{X}}^k) \leq M_1 \qquad a.s.\ on\ \Omega \tag{3.19}$$

$$\|\boldsymbol{x}_i^k\| \leq M_2 \qquad a.s.\ on\ \Omega \tag{3.20}$$

$$\|\boldsymbol{x}^k\| \leq M_3 \qquad a.s.\ on\ \Omega \tag{3.21}$$

Proof. By (3.17) and 4,

$$R(\boldsymbol{\mathcal{X}}^k) = \sqrt{\frac{1}{\lambda}\left(\frac{1}{m_k}\sum_{l=1}^{m_k}\|\mathcal{X}^{k,l}\|^2\right)} \leq \frac{M}{\sqrt{\lambda}} =: M_1.$$

Lemma 2 then implies

$$\|\boldsymbol{x}_i^k\| \leq \max\{\|\boldsymbol{x}_i^1\|, M_R\} =: M_2$$

and hence

$$\|\boldsymbol{x}^k\| = \sqrt{\sum_{i=1}^{p} \|\boldsymbol{x}_i^k\|^2} \leq \sqrt{p} M_{\boldsymbol{x}_i} =: M_3.$$

□

3.3.2 Regularity Estimates

In the following we exploit the multinomial structure of the cost functional to establish componentwise regularity estimates, such as local Lipschitz continuity of the sampled gradient $\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}, \mathcal{X})$ and of the Newton step $-\left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})\right)^{-1} \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})$, as well as bounded invertibility of the Hessian $\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})$. Note that since the mapping $\mathcal{X} \mapsto \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})$ is linear and the Hessian matrix deterministic, these estimates also hold when $f(\boldsymbol{x}; \mathcal{X})$ is replaced by the sample average $\tilde{f}(\boldsymbol{x}; \mathcal{X})$.

The estimates established in this section all follow from the local Lipschitz continuity of the mapping $\boldsymbol{x}_i^* \mapsto \Theta_i$, shown in 3 below.

Lemma 3. *Let Θ_i and $\tilde{\Theta}_i$ be matrices defined in terms of $\boldsymbol{x}_{i^*}$ and $\tilde{\boldsymbol{x}}_{i^*}$ respectively via* (3.6)*, where $\boldsymbol{x}_{i^*}, \tilde{\boldsymbol{x}}_{i^*}$ satisfy eq.* (3.20)*. Then there exists a constant C, independent of $\boldsymbol{x}_{i^*}$, $\tilde{\boldsymbol{x}}_{i^*}$, and $\mathcal{X}$, so that*

$$\|\Theta_i - \tilde{\Theta}_i\| \leq C \|\boldsymbol{x}_{i^*} - \tilde{\boldsymbol{x}}_{i^*}\|. \tag{3.22}$$

Proof. This result follows directly from the observation that the mapping $\boldsymbol{x}_{i^*} \mapsto \Theta_i$ is multilinear and hence smooth in $\boldsymbol{x}_{i^*}$. As a consequence, it is Lipschitz continuous over the closed convex set specified in eq. (3.20), with possible Lipschitz constant being the maximum of its gradient norm. □

Corollary 2. *Let $\boldsymbol{x}_{i^*}$ and $\tilde{\boldsymbol{x}}_{i^*}$ satisfy 3.20 with associated matrices Θ_i and $\tilde{\Theta}_i$ given by* (3.6)*. Then there is a constant $C \geq 0$ independent of $\boldsymbol{x}, \tilde{x}$, and $\mathcal{X}$ so that*

$$\|\Theta_i^T \Theta_i - \tilde{\Theta}_i^T \tilde{\Theta}_i\| \leq C \|\boldsymbol{x}_{i^*} - \tilde{\boldsymbol{x}}_{i^*}\|. \tag{3.23}$$

Proof. By the equivalence of the induced Euclidean and the Frobenius norms, there are constants C_1 and C_2 so that

$$\begin{aligned}\|\Theta_i^T\Theta_i - \tilde{\Theta}_i^T\tilde{\Theta}_i\| &\leq C_1\|\Theta_i^T\Theta_i - \tilde{\Theta}_i^T\tilde{\Theta}_i\|_2 = C_1\|\Theta_i^T(\Theta_i - \tilde{\Theta}_i) + (\Theta_i - \tilde{\Theta}_i)^T\tilde{\Theta}_i\|_2 \\ &\leq C_2\left(\|\Theta_i^T\|\|\Theta_i - \tilde{\Theta}_i\| + \|\Theta_i - \tilde{\Theta}_i\|\|\tilde{\Theta}_i\|\right).\end{aligned}$$

The result now follows from 3 and the bound 3.20. □

Letting $\tilde{\Theta}_i = 0$ in (3.23), yields the bound

$$\|\Theta_i^T\Theta_i\| \leq C_{\Theta^T\Theta}\|\boldsymbol{x}_{i^*}\|^{2(p-1)}. \tag{3.24}$$

As a first consequence of lemma 3 and corollary 2, we can obtain an explicit form for the componentwise Lipschitz constant of $\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})$.

Lemma 4. *For any $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{rn}$ satisfying 3.21, there exists a constant $C \geq 0$, independent of $\boldsymbol{x}$, $\tilde{x}$, and $\mathcal{X}$, so that*

$$\|\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) - \nabla_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}; \mathcal{X})\| \leq C\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|. \tag{3.25}$$

Proof. Let Θ_i and $\tilde{\Theta}_i$ be constructed from $\boldsymbol{x}_{i^*}$ and $\tilde{\boldsymbol{x}}_{i^*}$ respectively via 3.6. Using 3.7, the difference in sampled componentwise gradients is given by

$$\begin{aligned}&\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) - \nabla_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}; \mathcal{X}) \\ &= \left((\Theta_i^T - \tilde{\Theta}_i^T) \otimes I_{n_i}\right)\mathrm{vec}(\mathcal{X}_{(i)}) + (\Theta_i^T\Theta_i + \lambda I_r) \otimes I_{n_i}\boldsymbol{x}_i - (\tilde{\Theta}_i^T\tilde{\Theta}_i + \lambda I_r) \otimes I_{n_i}\tilde{\boldsymbol{x}}_i \\ &= \left((\Theta_i^T - \tilde{\Theta}_i^T) \otimes I_{n_i}\right)\mathrm{vec}(\mathcal{X}_{(i)}) + (\tilde{\Theta}_i^T\tilde{\Theta}_i + \lambda I_r) \otimes I_{n_i}(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i) \\ &\quad + (\Theta_i^T\Theta_i - \tilde{\Theta}_i^T\tilde{\Theta}_i) \otimes I_{n_i}\boldsymbol{x}_i.\end{aligned}$$

Since the singular values of a Kronecker product are formed from products of singular values of the constituent matrices, the matrix norm $\|B \otimes I\|_2 = \|B\|_2 \leq \|B\|$ for any matrix B. We therefore have

$$\begin{aligned} &\left\| \left((\Theta_i^T - \tilde{\Theta}_i^T) \otimes I_{n_i} \right) \mathrm{vec}(\mathcal{X}_{(i)}) + (\Theta_i^T \Theta_i - \tilde{\Theta}_i^T \tilde{\Theta}_i) \otimes I_{n_i} \boldsymbol{x}_i \right\| \\ &\leq \|\Theta_i - \tilde{\Theta}_i\|_2 \|\mathcal{X}\| + \|\Theta_i^T \Theta_i - \tilde{\Theta}_i^T \tilde{\Theta}_i\|_2 \|\boldsymbol{x}_i\| \end{aligned}$$

and

$$\|(\tilde{\Theta}_i^T \tilde{\Theta}_i + \lambda I_r) \otimes I_{n_i} (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)\| \leq \|\tilde{\Theta}_i^T \tilde{\Theta}_i + \lambda I_r\|_2 \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|.$$

The result now follows from Assumption 4, Lemma 3, and Corollary 2. □

Corollary 2 also implies the following uniform bounds on the componentwise Hessian.

Lemma 5. *For any $\boldsymbol{x} = (\boldsymbol{x}_i, \boldsymbol{x}_{i^*}) \in \mathbb{R}^{rn}$ satisfying the bound in (3.21) and any $\boldsymbol{v}_i \in \mathbb{R}^{rn_i}$, there is a constant $\lambda_{\max}$ independent of $\boldsymbol{x}, \boldsymbol{v}$, and $\mathcal{X}$, so that $0 < \lambda < \lambda_{\max} < \infty$ and*

$$\lambda \|\boldsymbol{v}_i\|^2 \leq \boldsymbol{v}_i^T \nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}) \boldsymbol{v}_i \leq \lambda_{\max} \|\boldsymbol{v}_i\|^2. \tag{3.26}$$

Proof. This follows directly from Corollary 2 and the fact that

$$\lambda \leq \|\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})\|_2 \leq \|\Theta_i^T \Theta_i\| + \lambda.$$

□

Finally, we establish the local Lipschitz continuity of the map from the current iterate to the Newton step, i.e $\boldsymbol{x} \mapsto (\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}))^{-1} \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})$.

Lemma 6. *For any $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{rn}$ satisfying* (3.21)*, there is a constant C independent of $\boldsymbol{x}, \tilde{\boldsymbol{x}}$ and $\mathcal{X}$ so that*

$$\|(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}))^{-1} \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) - (\nabla^2_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}))^{-1} \nabla_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}; \mathcal{X})\| \leq C \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|. \tag{3.27}$$

Proof. Let $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^{rn}$. By adding and subtracting a cross-term, we obtain

$$\begin{aligned} & \|(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}))^{-1} \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) - (\nabla^2_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}_i))^{-1} \nabla_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}; \mathcal{X})\| \\ \leq \; & \|(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}))^{-1} - (\nabla^2_{\tilde{\boldsymbol{x}}_i} f(\tilde{\boldsymbol{x}}))^{-1}\|_2 \|\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})\| \\ & + \|(\nabla^2_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}))^{-1}\|_2 \|\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) - \nabla_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}; \mathcal{X})\|. \end{aligned}$$

By the second resolvent identity (see e.g. Theorem 4.8.2. in [27]), we have

$$\begin{aligned} (\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}))^{-1} - (\nabla^2_{\tilde{\boldsymbol{x}}_i} f(\tilde{\boldsymbol{x}}))^{-1} &= (\Theta_i^T \Theta_i + \lambda I_r)^{-1} - (\tilde{\Theta}_i^T \tilde{\Theta}_i + \lambda I_r)^{-1} \\ &= (\Theta_i^T \Theta_i + \lambda I_r)^{-1} (\tilde{\Theta}_i \tilde{\Theta}_i - \Theta_i^T \Theta_i)(\tilde{\Theta}_i^T \tilde{\Theta}_i + \lambda I_r)^{-1}, \end{aligned}$$

so that, by virtue of Corollary 2 and lemmas 4 and 6, there is a constant C_1 for which

$$\|(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}))^{-1} - (\nabla^2_{\tilde{\boldsymbol{x}}_i} f(\tilde{\boldsymbol{x}}))^{-1}\|_2 \|\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X})\| \leq \frac{C_1}{\lambda} \|\boldsymbol{x}_{i^*} - \tilde{\boldsymbol{x}}_{i^*}\|.$$

Moreover, Lemmas 4 and 5 imply

$$\|(\nabla^2_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}))^{-1}\|_2 \|\nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \mathcal{X}) - \nabla_{\boldsymbol{x}_i} f(\tilde{\boldsymbol{x}}; \mathcal{X})\| \leq \frac{C_2}{\lambda} \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|.$$

Combining these estimates gives the bound in eq. (3.27). □

3.3.3 Convergence

We now consider the difference $\boldsymbol{\delta}^k = \tilde{\boldsymbol{g}}^k - \boldsymbol{g}^k$ between the sampled and expected search directions. For the standard stochastic gradient descent algorithm, this stochastic quantity vanishes in expectation, given past information $\mathcal{F}^{k-1} = \sigma(\boldsymbol{\mathcal{X}}^1, ..., \boldsymbol{\mathcal{X}}^{k-1})$, i.e. $\mathbb{E}\left[\boldsymbol{\delta}^k | \mathcal{F}^{k-1}\right] = 0$, since $\tilde{\boldsymbol{g}}^k$ is an unbiased estimator of $\boldsymbol{g}^k$. For the stochastic alternating least squares method, this is no longer the case. Lemma 7 however uses the regularity of the gradient and the Hessian to establish an upper bound that decreases on the order $O(\frac{1}{k})$ as $k \to \infty$.

Lemma 7. *There is a constant $C \geq 0$ such that*

$$\left\| \mathbb{E}\left[(H^{k,i})^{-1} \boldsymbol{\delta}^{k,i} | \mathcal{F}^{k-1} \right] \right\| \leq \frac{C}{k}, \qquad \text{for } i = 1, ..., p, \; k = 1, 2, ... \tag{3.28}$$

Proof. Recall that the current iterate $\boldsymbol{x}^{k,i}$ is statistically dependent on sampled tensors $\boldsymbol{\mathcal{X}}^1, \ldots, \boldsymbol{\mathcal{X}}^{k-1}$, while its first i components also depend on $\boldsymbol{\mathcal{X}}^k$. For the purpose of computing $\mathbb{E}\left[(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} | \mathcal{F}^{k-1}\right]$, we suppose that $\boldsymbol{\mathcal{X}}^1, \ldots, \boldsymbol{\mathcal{X}}^{k-1}$ are known and write $\boldsymbol{x}^{k,i-1} = \boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k) = (\boldsymbol{x}_1^{k+1}(\boldsymbol{\mathcal{X}}^k), \ldots, \boldsymbol{x}_{i-1}^{k+1}(\boldsymbol{\mathcal{X}}^k), \boldsymbol{x}_i^k, \ldots, \boldsymbol{x}_p^k)$ to emphasize its dependence on $\boldsymbol{\mathcal{X}}^k$. Thus

$$\begin{aligned}
&\mathbb{E}\left[(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} | \mathcal{F}^{k-1}\right] = \\
&\int_{\Omega^{m_k}} \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k))\right)^{-1} \left(\nabla_{\boldsymbol{x}_i} \tilde{f}(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k); \boldsymbol{\mathcal{X}}^k) - \nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k))\right) d\mu_{\boldsymbol{\mathcal{X}}^k}.
\end{aligned}$$

By definition, and since $\nabla_{\boldsymbol{x}_i}\tilde{f}$ is an unbiased estimator of $\nabla_{\boldsymbol{x}_i} F$, we have

$$\begin{aligned}
\nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k)) &= \int_{\Omega} \nabla_{x_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k); \mathcal{X}) d\mu_{\mathcal{X}} \\
&= \int_{\Omega^{m_k}} \nabla_{x_i} \tilde{f}(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k); \boldsymbol{\mathcal{X}}) d\mu_{\boldsymbol{\mathcal{X}}}.
\end{aligned}$$

Moreover, since $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}}^k$ are identically distributed,

$$\begin{aligned}
&\int_{\Omega^{m_k}} \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k))\right)^{-1} \nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k)) d\mu_{\boldsymbol{\mathcal{X}}^k} \\
&= \int_{\Omega^{m_k}} \int_{\Omega^{m_k}} \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k))\right)^{-1} \nabla_{\boldsymbol{x}_i} \tilde{f}(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k); \boldsymbol{\mathcal{X}}) d\mu_{\boldsymbol{\mathcal{X}}} d\mu_{\boldsymbol{\mathcal{X}}^k} \\
&= \int_{\Omega^{m_k}} \int_{\Omega^{m_k}} \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}))\right)^{-1} \nabla_{\boldsymbol{x}_i} \tilde{f}(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}); \boldsymbol{\mathcal{X}}^k) d\mu_{\boldsymbol{\mathcal{X}}} d\mu_{\boldsymbol{\mathcal{X}}^k}.
\end{aligned}$$

Therefore

$$\begin{aligned}
&\mathbb{E}\left[(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} | \mathcal{F}^{k-1}\right] \\
&\qquad = \int_{\Omega^{m_k}} \int_{\Omega^{m_k}} \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k))\right)^{-1} \nabla_{\boldsymbol{x}_i} \tilde{f}(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k); \boldsymbol{\mathcal{X}}^k) d\mu_{\boldsymbol{\mathcal{X}}} d\mu_{\boldsymbol{\mathcal{X}}^k} \\
&\qquad\quad - \int_{\Omega^{m_k}} \int_{\Omega^{m_k}} \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}))\right)^{-1} \nabla_{\boldsymbol{x}_i} \tilde{f}(\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}); \boldsymbol{\mathcal{X}}^k) d\mu_{\boldsymbol{\mathcal{X}}} d\mu_{\boldsymbol{\mathcal{X}}^k}. \quad (3.29)
\end{aligned}$$

In the special case $i = 1$, the iterate $\boldsymbol{x}^{k,0} = \boldsymbol{x}^{k-1}$, and hence $\boldsymbol{x}^{k,1}$ does not depend on $\boldsymbol{\mathcal{X}}^k$. Since $\nabla_{\boldsymbol{x}_1} \tilde{f}(\boldsymbol{x}^{k-1}; \boldsymbol{\mathcal{X}}^k)$ is an unbiased estimator of $\nabla_{\boldsymbol{x}_1} F(\boldsymbol{x}^{k-1})$, we have

$$\mathbb{E}\left[(H^{k,1})^{-1}\boldsymbol{\delta}^{k,1} | \mathcal{F}^{k-1}\right] = 0.$$

We now consider the case $i = 2, ..., p$. Using the Lipschitz continuity of the mapping $\boldsymbol{x} \mapsto \left(\nabla^2_{\boldsymbol{x}_i} f(\boldsymbol{x})\right)^{-1} \nabla_{\boldsymbol{x}_i} f(\boldsymbol{x}; \boldsymbol{\mathcal{X}}^k)$ (lemma 6), the bounds in Corollary 1, and Jensen's inequality,

$$\left\|\mathbb{E}\left[(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} | \mathcal{F}^{k-1}\right]\right\| \leq C_1 \int_{\Omega^{m_k}} \int_{\Omega^{m_k}} \|\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k) - \boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}})\| d\mu_{\boldsymbol{\mathcal{X}}} d\mu_{\boldsymbol{\mathcal{X}}^k},$$

for some constant C_1. The integrand above can be bounded by

$$\begin{aligned}\|\boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}}^k) - \boldsymbol{x}^{k,i-1}(\boldsymbol{\mathcal{X}})\| &\leq \sum_{j=1}^{i-1} \|\boldsymbol{x}_j^{k+1}(\boldsymbol{\mathcal{X}}^k) - \boldsymbol{x}_j^{k+1}(\boldsymbol{\mathcal{X}})\| \\ &= \sum_{j=1}^{i-1} \alpha^{k,j} \|(H^{k,j}(\boldsymbol{\mathcal{X}}^k))^{-1}\tilde{\boldsymbol{g}}^{k,j}(\boldsymbol{\mathcal{X}}^k) - (H^{k,j}(\boldsymbol{\mathcal{X}}))^{-1}\tilde{\boldsymbol{g}}^{k,j}(\boldsymbol{\mathcal{X}})\| \\ &\leq \frac{c_{\max}}{k}\left(\|(H^{k,j}(\boldsymbol{\mathcal{X}}^k))^{-1}\tilde{\boldsymbol{g}}^{k,j}(\boldsymbol{\mathcal{X}}^k)\| + \|(H^{k,j}(\boldsymbol{\mathcal{X}}))^{-1}\tilde{\boldsymbol{g}}^{k,j}(\boldsymbol{\mathcal{X}})\|\right).\end{aligned}$$

The result now follows from taking expectations and using (3.27) with $\tilde{\boldsymbol{x}} = \boldsymbol{0}$, in conjunction with (3.19), and (3.21). □

Lemma 8. *If $\boldsymbol{h}$ is $\mathcal{F}^{k-1}$-measurable and $\mathbb{E}[\|\boldsymbol{h}\|] < \infty$ then there is a constant $0 \leq C < \infty$*

$$\mathbb{E}\left[\left\langle \boldsymbol{h}, (H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\right\rangle\right] \leq \frac{C}{k}\mathbb{E}\left[\|\boldsymbol{h}\|\right]. \tag{3.30}$$

Proof. Using the law of total expectation, the $\mathcal{F}^{k-1}$-measurability of $\boldsymbol{h}$, and lemma 7, we have

$$\begin{aligned}\mathbb{E}\left[\left\langle \boldsymbol{h}, (H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\right\rangle\right] &= \mathbb{E}_{\boldsymbol{\mathcal{X}}^1,\ldots,\boldsymbol{\mathcal{X}}^{k-1}}\left[\mathbb{E}\left[\left\langle \boldsymbol{h}, (H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\right\rangle \middle| \mathcal{F}^{k-1}\right]\right] \\ &= \mathbb{E}_{\boldsymbol{\mathcal{X}}^1,\ldots,\boldsymbol{\mathcal{X}}^{k-1}}\left[\left\langle \mathbb{E}\left[\boldsymbol{h}\middle|\mathcal{F}^{k-1}\right], \mathbb{E}\left[(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\middle|\mathcal{F}^{k-1}\right]\right\rangle\right] \\ &\leq \mathbb{E}_{\boldsymbol{\mathcal{X}}^1,\ldots,\boldsymbol{\mathcal{X}}^{k-1}}\left[\|\mathbb{E}\left[\boldsymbol{h}\right]\| \left\|\mathbb{E}\left[(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\middle|\mathcal{F}^{k-1}\right]\right\|\right] \\ &\leq \mathbb{E}_{\boldsymbol{\mathcal{X}}^1,\ldots,\boldsymbol{\mathcal{X}}^{k-1}}\left[\frac{C}{k}\mathbb{E}_{\boldsymbol{\mathcal{X}}^k}[\|\boldsymbol{h}\|]\right] = \frac{C}{k}\mathbb{E}[\|\boldsymbol{h}\|].\end{aligned}$$

□

The main convergence theorem is based on the following lemma (for a proof, see e.g. Lemma A.5, [40])

Lemma 9. *Let $\{a_k\}_{k=1}^{\infty}$ and $\{b_k\}_{k=1}^{\infty}$ be any two nonnegative, real sequences so that (i) $\sum_{k=1}^{\infty} a_k = \infty$, (ii) $\sum_{k=1}^{\infty} a_k b_k < \infty$, and (iii) there is a constant $K > 0$ so that $|b_{k+1} - b_k| \leq Ka_k$ for $k \geq 1$. Then $\lim_{k\to\infty} b_k = 0$.*

Theorem 3.1. *

Proof. We base the proof on lemma 9 with $a_k = \frac{1}{k}$ and $b_k = \|\boldsymbol{g}^{k,i}\|^2$. Clearly, Condition (i) in lemma 9 is satisfied. To show that Condition (ii) holds, i.e. that $\sum_{k=1}^{\infty} \frac{1}{k}\mathbb{E}\left[\|\boldsymbol{g}^{k,i}\|^2\right] < \infty$ for $i = 1, \ldots, p$, we use the componentwise Taylor expansion (3.13) of the expected cost centered at the iterate $\boldsymbol{x}^{k,i-1}$ and the SALS update given in (3.15) to express the expected decrease as

$$
\begin{aligned}
& F(\boldsymbol{x}^{k,i}) - F(\boldsymbol{x}^{k,i-1}) \\
=& \nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i})^T(\boldsymbol{x}^{k,i+1} - \boldsymbol{x}^{k,i}) + \frac{1}{2}(\boldsymbol{x}^{k,i+1} - \boldsymbol{x}^{k,i})^T \nabla^2_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i})(\boldsymbol{x}^{k,i+1} - \boldsymbol{x}^{k,i}) \\
=& -\alpha^{k,i}(\boldsymbol{g}^{k,i})^T(H^{k,i})^{-1}\tilde{\boldsymbol{g}}^{k,i} + \frac{1}{2}(\alpha^{k,i})^2((H^{k,i})^{-1}\tilde{\boldsymbol{g}}^{k,i})^T H^{k,i}(H^{k,i})^{-1}\tilde{\boldsymbol{g}}^{k,i} \\
=& -\alpha^{k,i}\boldsymbol{g}^{k,i^T}(H^{k,i})^{-1}\boldsymbol{g}^{k,i} - \alpha^{k,i}(\boldsymbol{g}^{k,i})^T(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} + \frac{1}{2}(\alpha^{k,i})^2(\tilde{\boldsymbol{g}}^{k,i})^T(H^{k,i})^{-1}\tilde{\boldsymbol{g}}^{k,i}.
\end{aligned}
$$

Recall from eq. (3.16) that $\alpha^{k,i} = \frac{c^{k,i}}{k}$, with $0 < c_{\min} \leq c^{k,i} \leq c_{\max} \leq 2$. Since, by (3.20) and lemma 5,

$$
\alpha^{k,i}(\boldsymbol{g}^{k,i})^T(H^{k,i})^{-1}\boldsymbol{g}^{k,i} \geq \frac{c_{\min}}{k}\frac{1}{\lambda_{\max}}\|\boldsymbol{g}^{k,i}\|^2,
$$

the above expression can be rearranged as

$$
\begin{aligned}
\frac{1}{k}\|\boldsymbol{g}^{k,i}\|^2 \leq \frac{\lambda_{\max}}{c_{\min}}\left(E_1^{k,i} + E_2^{k,i} + E_3^{k,i}\right), \quad \text{with} \qquad (3.31) \\
E_1^{k,i} = F(\boldsymbol{x}^{k,i-1}) - F(\boldsymbol{x}^{k,i}), \\
E_2^{k,i} = -\alpha^{k,i}(\boldsymbol{g}^{k,i})^T(\tilde{H}^{k,i})^{-1}\boldsymbol{\delta}^{k,i}, \text{ and} \\
E_3^{k,i} = \frac{1}{k^2}\frac{c_{\max}}{2}(\tilde{\boldsymbol{g}}^{k,i})^T(H^{k,i})^{-1}\tilde{\boldsymbol{g}}^{k,i}.
\end{aligned}
$$

Evidently, Condition (ii) of lemma 9 holds as long as $\sum_{k=1}^{\infty} \mathbb{E}\left[E_j^{k,i}\right] < \infty$ for $j = 1, 2, 3$. For a fixed $K > 0$, the first term $E_1^{k,i}$ generates a telescoping sum so that

$$\begin{aligned}\sum_{k=1}^{K}\sum_{i=1}^{p} \mathbb{E}\left[E_1^{k,i}\right] &= \sum_{k=1}^{K} \mathbb{E}\left[F(\boldsymbol{x}^k)\right] - \mathbb{E}\left[F(\boldsymbol{x}^{k+1})\right] \\ &= \mathbb{E}\left[F(\boldsymbol{x}^1)\right] - \mathbb{E}\left[F(\boldsymbol{x}^{K+1})\right] \leq \mathbb{E}\left[F(\boldsymbol{x}^1)\right] < \infty \qquad \forall K > 0, \end{aligned} \tag{3.32}$$

since $F(\boldsymbol{x}^{K+1}) \geq 0$. Consider

$$\begin{aligned} E_2^{k,i} &= -\alpha^{k,i}(\boldsymbol{g}^{k,i})^T(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} \\ &= -\alpha^{k,i}(\boldsymbol{g}^{k,i} - \nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^k))^T(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} + \alpha^{k,i}\nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^k)^T(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}. \end{aligned}$$

Since $\nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^k)$ is $\mathcal{F}^{k-1}$-measurable, lemma 8 implies that the expectation of the second term can be bounded as follows

$$\mathbb{E}\left[\alpha^{k,i}\nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^k)^T(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\right] \leq \frac{C_1}{k^2},$$

for an appropriate constant $C_1 \geq 0$ guaranteed by the regularity of the gradient and boundedness of the iterates, i.e. lemma 4 and corollary 1. The first term satisfies

$$\begin{aligned} &- \alpha^{k,i}(\nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^{k,i}) - \nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^k))^T(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i} \\ &\leq \alpha^{k,i}\|\nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^{k,i}) - \nabla_{\boldsymbol{x}_i}F(\boldsymbol{x}^k)\|\|(H^{k,i})^{-1}\boldsymbol{\delta}^{k,i}\| \\ &\leq C_2\alpha^{k,i}\|\boldsymbol{x}^{k,i} - \boldsymbol{x}^k\| \\ &= C_2\alpha^{k,i}\sqrt{\sum_{j=1}^{i}(\alpha^{k,j})^2\|(H^{k,j})^{-1}\tilde{\boldsymbol{g}}^{k,j}\|^2} \leq \frac{C_3}{k^2}, \end{aligned}$$

where C_2 and C_3 are constants given by the regularity of the gradient (Lemma 4) and that of the Newton step (Lemma 6) respectively. Combining these two bounds yields,

$$\sum_{k=1}^{\infty}\sum_{i=1}^{p}\mathbb{E}\left[E_2^{k,i}\right] \leq C_4 \sum_{k=1}^{\infty}\frac{1}{k^2} < \infty, \tag{3.33}$$

for an appropriate constant $C_4 \geq 0$. Finally, we use the regularity of the Newton step and of the gradient, together with the iterate bound, i.e. Lemmas 4 and 6 and corollary 1, to bound

$$\mathbb{E}\left[(\tilde{\boldsymbol{g}}^{k,i})^T (H^{k,i})^{-1}\tilde{\boldsymbol{g}}^{k,i}\right] \leq C_5,$$

for some constant C_5 so that

$$\sum_{k=1}^{\infty}\sum_{i=1}^{p}\mathbb{E}\left[E_3^{k,i}\right] \leq \frac{c_{\max}}{2}\sum_{k=1}^{\infty}\sum_{i=1}^{p}\frac{C_5}{k^2} < \infty. \tag{3.34}$$

By virtue of Inequality (3.31), the upper bounds (3.32), (3.33), and (3.34) now imply that Condition (ii) of lemma 9 holds. It now remains to show that Condition (iii) of lemma 9 holds, i.e. that $\left|\mathbb{E}\left[\|\boldsymbol{g}^{k+1,i}\|^2\right] - \mathbb{E}\left[\|\boldsymbol{g}^{k,i}\|^2\right]\right| = O(1/k)$ as $k \to \infty$ for all $i = 1, \ldots, p$. By the reverse triangle inequality, the Lipschitz continuity of the gradient and the Newton step and the boundedness of the iterates, there are constants $C_6, C_7 \geq 0$ so that

$$\begin{aligned}
&\|\boldsymbol{g}^{k+1,i}\|^2 - \|\boldsymbol{g}^{k,i}\|^2 \leq (\|\boldsymbol{g}^{k+1,i}\| + \|\boldsymbol{g}^{k,i}\|)(\|\boldsymbol{g}^{k+1,i} - \boldsymbol{g}^{k,i}\|) \leq C_6\|\boldsymbol{x}^{k+1,i} - \boldsymbol{x}^{k,i}\| \\
&= C_6\sqrt{\sum_{j=1}^{i}\|\boldsymbol{x}_j^{k+2} - \boldsymbol{x}_j^{k+1}\|^2 + \sum_{j=i+1}^{p}\|\boldsymbol{x}_j^{k+1} - \boldsymbol{x}_j^{k}\|^2} \\
&= C_6\sqrt{\sum_{j=1}^{i}\left(\alpha^{k+1,j}\right)^2\|(H^{k+1,j})^{-1}\tilde{\boldsymbol{g}}^{k+1,j}\|^2 + \sum_{j=i+1}^{p}\left(\alpha^{k,j}\right)^2\|(H^{k,j})^{-1}\tilde{\boldsymbol{g}}^{k,j}\|^2} \\
&\leq C_6\frac{c_{\max}}{k}\sqrt{\sum_{j=1}^{i}\|(H^{k+1,j})^{-1}\tilde{\boldsymbol{g}}^{k+1,j}\|^2 + \sum_{j=i+1}^{p}\|(H^{k,j})^{-1}\tilde{\boldsymbol{g}}^{k,j}\|^2} \leq \frac{C_7}{k}.
\end{aligned}$$

□

Theorem 3.1 implies directly that the full gradient

$$\nabla F(\boldsymbol{x}^{k,i}) = [\nabla_{\boldsymbol{x}_1} F(\boldsymbol{x}^{k,i}), ..., \nabla_{\boldsymbol{x}_p} F(\boldsymbol{x}^{k,i})]^T$$

converges to zero in mean square. Indeed, for any tolerance $\varepsilon > 0$, choose k_0 large enough so that $\mathbb{E}[\|\nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i})\|^2] < \varepsilon/p$ for all $i = 1, ..., p$, whenever $k \geq k_0$. For these values of k,

$$\mathbb{E}[\|\nabla F(\boldsymbol{x}^{k,i})\|^2] = \sum_{i=1}^{p} \mathbb{E}[\|\nabla_{\boldsymbol{x}_i} F(\boldsymbol{x}^{k,i})\|^2] < \varepsilon.$$

Moreover, Jensen's inequality implies that $\mathbb{E}[\nabla F(\boldsymbol{x}^{k,i})] \to 0$ as $k \to \infty$. Our result does, however, not guarantee the convergence of the iterates themselves. At most, we can conclude that the limits of all convergent subsequences of iterates $\boldsymbol{x}^{k,i}$, which exist by virtue of assumption 4, converge to stationary points.

3.4 Numerical Experiments

In this section we detail the implementation of the SALS algorithm, discuss its computational complexity, and perform two numerical experiments to validate our theoretical results.

We invoke the matricized form (3.9) of the componentwise minimization problem for the sake of computational and storage efficiency. In particular, at the k-th block iterate, we cycle through each component $i = 1, \ldots, p$, computing the componentwise minimizer as the stationary matrix $\hat{A}_i^{k+1}$ satisfying the matrix equation

$$\tilde{\mathcal{X}}_{(i)}^{k} \Theta_i = \hat{A}_i^{k+1} (\Theta_i^T \Theta_i + \lambda I_r), \tag{3.35}$$

where $\tilde{\mathcal{X}}^k = \frac{1}{m_k} \sum_{l=1}^{m_k} \mathcal{X}^{k,l}$ is the sample/batch tensor average. Note that the product $\tilde{\mathcal{X}}_{(i)}^{k} \Theta_i$ can be computed in parallel as the average of sample products $\mathcal{X}_{(i)}^{k,l} \Theta_i$. Finally, we update

the i-th component factor matrix A_i^k using the relaxation step

$$A_i^{k+1} = \alpha^{k,i}\hat{A}_i^{k+1} + (1-\alpha^{k,i})A_i^k.$$

In our numerical experiments we use stepsizes of the form $\alpha^{k,i} = \frac{\tau}{k}$ for $i = 1,\ldots,p$ and $k = 1,2,\ldots$, where $\tau \in (0,2]$. We have also found that using a constant stepsize $\tau = 1$ during the initial iterations (a burn-in period) can lead to a significant reduction in the residual. We compare the performance of the SALS method with that of the stochastic gradient descent (SGD) method, which was investigated in [34] for a broad class of loss functions and shown to offer significant speedup over its deterministic counterparts. In the case of Problem 3.2, these algorithms use the update step

$$A_i^{k+1} = A_i^k - \alpha^k g_i(A_1^k,\ldots,A_p^k), \qquad i = 1,...,p,$$

where g_i denotes the partial gradient

$$g_i(A_1^k,\ldots,A_p^k) = -\tilde{\mathcal{X}}_{(i)}^k\Theta_i + A_i^k(\Theta_i^T\Theta_i + \lambda I_r)$$

and α^k is an appropriate stepsize sequence. To make a fair comparison, we use either the Robbins-Monro [52] stepsize rule $\alpha_k = \frac{\tau}{k}$ or a constant stepsize, whichever gives better results. The SGD update is very sensitive to the stepsize parameter τ, which must therefore be callibrated to each problem. While there are many variations of stochastic optimization methods in the use of stepsize rules, such as adaptive moment estimation (Adam [32]), convergence criteria [57], sub-sampling strategies [34], the use of sketching methods [6] to solve Equation (3.35), and parallelization, a detailed study and comparison of these is beyond the scope of the current work.

We implementated our method in Matlab using the Tensor Toolbox [35]. For each of the numerical examples, we constructed a random tensor whose expectation $\mathbb{E}[\mathcal{X}]$ has a known

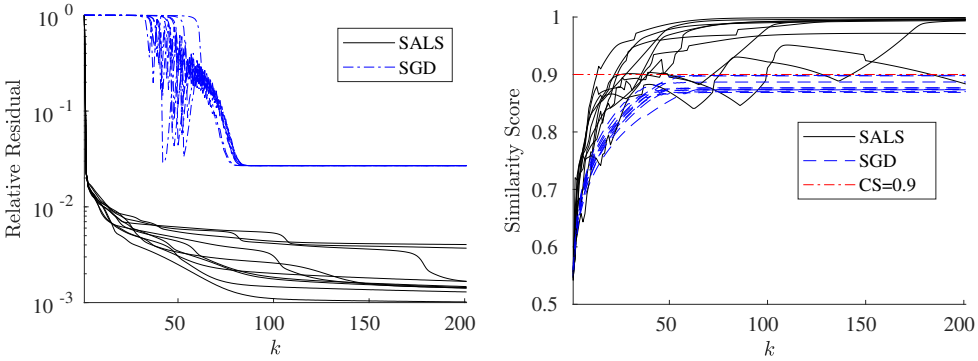
decomposition and assessed the accuracy of the reconstruction by means of the relative error $\|\mathbb{E}[\mathcal{X}] - [\![\boldsymbol{x}]\!]\|/\|\mathbb{E}[\mathcal{X}]\|$ and the cosine similarity score of the factor matrices (see [34]), computed as

$$\frac{1}{r}\sum_{j=1}^{r}\prod_{i=1}^{p}\frac{\hat{\boldsymbol{a}}_{j,i}\cdot\boldsymbol{a}_{j,i}}{\|\hat{\boldsymbol{a}}_{j,i}\|\|\boldsymbol{a}_{j,i}\|},$$

where $A_i = [\boldsymbol{a}_{1,i}, ..., \boldsymbol{a}_{p,i}]$ and $\hat{A}_i = [\hat{\boldsymbol{a}}_{1,i}, ..., \hat{\boldsymbol{a}}_{p,i}]$ are the true and factor matrices respectively, with columns permuted to ensure the highest possible similarity. We consider a factor matrix recovered if the similarity score is above 0.9.

Example 3 (Random Tensor)**.** In our first example, we consider truly random tensors whose statistics are not known *a priori*, i.e. for which we cannot simply apply the ALS method to their expectation. We apply the SALS algorithm to a 4-dimensional dense random tensor of size $(30, 30, 30, 30)$ constructed from a deterministic tensor $\mathcal{X}^*$ with known decomposition of rank 10. The factor matrices $A_1, ..., A_4$ are initialized randomly with entries drawn independently from a $N(5, 1)$ distribution. During the iteration, we generate random samples $\mathcal{X}$ by perturbing each entry of $\mathcal{X}^*$ independently by a uniformly distributed noise $U \sim \text{UNIF}(-10, 10)$, so that $\mathbb{E}[\mathcal{X}] = \mathcal{X}^*$. This ensures sample tensors that are uniformly bounded, as required by Assumption 4. Both the SALS and SGD algorithms were run for 200 iterations with a regularization parameter of $\lambda = 10^{-10}$. The constant stepsize $\tau = 1$ was used in the SALS method, while the decreasing stepsize of $\frac{2.5\times10^{-9}}{k}$ was found to yield optimal convergence for the SGD method. Example 3 compares the convergence of the two methods for 10 runs with different initial conditions and tensor sample realizations. The entries of the factor matrices where initialized randomly and independently from uniform distributions UNIF(0,1). For each run the SALS method achieves a better convergence rate and exhibits fewer oscillations than the SGD method.

We observe a strong initial decrease in the residual for the SALS method, followed by a flattening out.

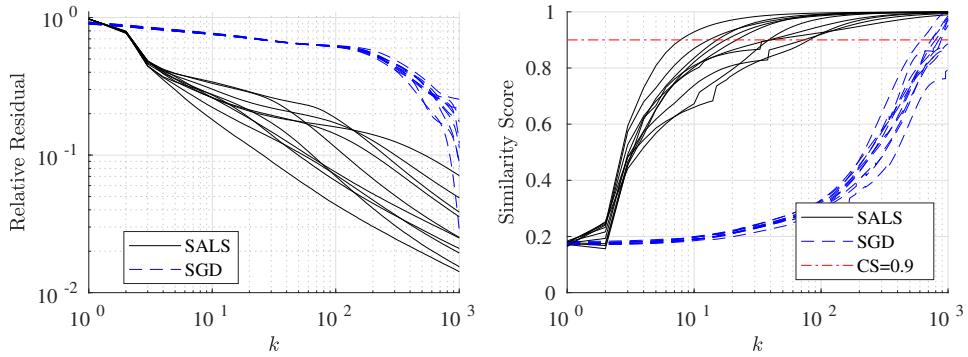


(a) Semi-log plot of the relative residual

(b) Plot of the cosine similarity score with the acceptible threshold in red.

Figure 3.1: Convergence plots of the SALS and SGD methods applied to Example 3, for 10 independent runs differing in initial guess and tensor samples.

Example 4 (Sub-Sampled Deterministic Tensor)**.** To test the SALS algorithm's convergence properties in the context of sub-sampling, we decompose a $50\times50\times50\times50$-dimensional dense tensor $\mathcal{X}$ with $N = 6.25 \times 10^6$ entries and a known decomposition of rank 10. The reference factor matrices A_1, A_2, A_3, and A_4 were initialized randomly with entries drawn from a $N(1, 1)$ distribution. To obtain tensor samples $\mathcal{X}^{k,l}$ we sub-sample 5% of $\mathcal{X}$ uniformly, i.e we choose $s = 312'000$ entries randomly with replacement. The number of times $s_{\boldsymbol{i}}$ the index $\boldsymbol{i}$ is chosen follows the binomial distribution, specifically $s_{\boldsymbol{i}} \sim \text{BIN}\left(s, \frac{1}{N}\right)$, and the resulting tensor entry $\mathcal{X}^{k,l}_{\boldsymbol{i}} = \frac{s_{\boldsymbol{i}}N}{s}\mathcal{X}_{\boldsymbol{i}}$ has mean $\mathbb{E}[\mathcal{X}^{k,l}_{\boldsymbol{i}}] = \mathcal{X}_{\boldsymbol{i}}$ and variance $\mathbb{E}\left[(\mathcal{X}^{k,l}_{\boldsymbol{i}} - \mathcal{X}_{\boldsymbol{i}})^2\right] = \mathcal{X}^2_{\boldsymbol{i}}\frac{(N-1)}{s}$ (see [34]). Note that Assumption 4 is automatically satisfied in this case. Both the SALS and SGD algorithms were run 10 times for 1000 iterations with a regularization parameter of $\lambda = 10^{-8}$. The stepsize rule $\alpha^{k,i} = \frac{1.8}{k}$ was used in the SALS method, while the decreasing stepsize of $\alpha^k = \frac{10^{-8}}{k}$ was found to yield optimal convergence for the SGD method. Example 4 compares the convergence of the two methods for 10 runs with different initial conditions and tensor sample realizations. The entries of the factor matrices where initialized randomly and independently from a UNIF(0,1) distritbution, as before. For each run the SALS method again achieves a better convergence rate than the SGD method, recovering the true factor matrix after 100 iterations, whereas SGD does so only after 1000 iterations. Again, the SALS method yields a strong initial decrease in the residual.



(a) Log-log plot of the relative residuals

(b) Plot of the cosine fit coefficients using log-scale for the iteration count.

Figure 3.2: Convergence plots for 10 independent runs of the SALS and SGD algorithms, differing in initial guess and sampling.

Computational Complexity

As mentioned earlier, the computational effort of each block iteration of the SALS algorithm is determined by the cost of forming and solving Equation (3.35) for each component. Recall that r is the specified rank, p the tensor's dimension, and $n = \sum_{i=1}^{p} n_i$ where n_i is the size of the i-th vector component of each outer product in the expansion. Further, let $N = \prod_{i=1}^{p} n_i$ be the total number of entries in $\mathcal{X}$ and $\text{nnz}(\mathcal{X})$ be the number of non-zero entries in $\mathcal{X}$. It can then readily be seen that the cost of forming the coefficient matrix $\Theta_i^T \Theta_i + \lambda I_r$ is $O(pr^2 n)$, that of computing the matricized tensor times Khatri-Rao product (MTTKRP) $\tilde{\mathcal{X}}_{(i)}^k \Theta_i$ is $O(pr \cdot \text{nnz}(\tilde{\mathcal{X}}_{(i)}^k))$, and that of solving the resulting dense symmetric system is $O(pr^3)$. If $r \ll N$, the computational cost of the MTTKRP dominates, especially as the density of $\tilde{\mathcal{X}}_{(i)}^k$ increases and hence $\text{nnz}(\tilde{\mathcal{X}}_{(i)}^k) \to N$. In comparison, the per-iteration cost for the SGD method is the same as that for the SALS method, except that it does not require p linear system solve, so that these methods differ in cost by $O(kpr^3)$, where k is the block iteration count.

3.5 Conclusion

Stochastic-gradient-based optimization approaches offer an efficient means of decomposing random or large sub-sampled tensors. In this work, we developed a convergence theory for the SALS method, showing that all accumulation points converge to stationary points in expectation. In our numerical experiments, the SALS method exhibited a strong initial decrease in relative residual, suggesting its potential to be used as an initial preconditioner in conjunction with other optimization approaches. In our analysis we focused on regularization in the Frobenius norm, and have not included a discussion on related proximal point algorithms or other regularization approaches (see e.g. [36, 56, 64]). Another interesting avenue of exploration relates to the choice of cost functional in tensor decomposition. Even though the quadratic structure afforded by the Frobenius norm is essential to the effectiveness of the ALS method, the standard statistical basis of comparison, namely expectation, could potentially be extended to other statistical metrics, such as those used in risk averse optimization methods. Finally, we foresee the application and extension of the SALS in the analysis and decomposition of other real datasets.

Chapter 4

Conclusion

In this dissertation, we see the AdaGrad method being applied to cost/loss function with strong convexity, Lipschitz continuity and regularity. A similar observation can be made to the application of the SALS method where the loss function is componentwise convex (quadratic to be specific) and regular in the Frobenius norm. Both methods have exhibited fast descent in the initial iterations of the algorithm. It needs to be seen if these methods can be applied to more complex problems with uncertainty.

The future projects would be to find applications of other stochastic-gradient-based algorithms to optimal control. With these tools in hand, it seems promising to be able to find applications of neural network to design control, being an optimization problem. Since neural networks rely on the use of tensors, we can combine the above two methods together where the design of the optimal control can be made cheaper using tensors made simpler through tensor decomposition.

Bibliography

[1] Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. "A scalable optimization approach for fitting canonical tensor decompositions". In: *Journal of Chemometrics* 25.2 (Jan. 2011), pp. 67–86. DOI: 10.1002/cem.1335.

[2] Alekh Agarwal et al. "Information-theoretic lower bounds on the oracle complexity of convex optimization". In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio et al. Curran Associates, Inc., 2009, pp. 1–9.

[3] Elizabeth S Allman and John A Rhodes. "The identifiability of tree topology for phylogenetic models, including covarion and mixture models". In: *Journal of Computational Biology* 13.5 (2006), pp. 1101–1113.

[4] Isaac Asimov. *I, robot*. New York: Bantam Books, 2004. ISBN: 978-0785773382.

[5] Ivo Babuška, Fabio Nobile, and Raúl Tempone. "A Stochastic Collocation Method for Elliptic Partial Differential Equations with Random Input Data". In: *SIAM Journal on Numerical Analysis* 45.3 (Jan. 2007), pp. 1005–1034. DOI: 10.1137/050645142. URL: https://doi.org/10.1137/050645142.

[6] Casey Battaglino, Grey Ballard, and Tamara G. Kolda. "A Practical Randomized CP Tensor Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 39.2 (Jan. 2018), pp. 876–901. DOI: 10.1137/17m1112303.

[7] Amir Beck and Luba Tetruashvili. "On the Convergence of Block Coordinate Descent Type Methods". In: *SIAM Journal on Optimization* 23.4 (Jan. 2013), pp. 2037–2060. DOI: 10.1137/120887679.

[8] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. USA: Prentice-Hall, Inc., 1989. ISBN: 0136487009.

[9] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. "Optimization Methods for Large-Scale Machine Learning". In: *SIAM Review* 60.2 (Jan. 2018), pp. 223–311. DOI: 10.1137/16m1080173.

[10] Guillaume Bouchard et al. "Matrix and Tensor Factorization Methods for Natural Language Processing". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Tutorial Abstracts*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 16–18. DOI: 10.3115/v1/P15-5005. URL: https://www.aclweb.org/anthology/P15-5005.

[11] M. Boussé, O. Debals, and L. De Lathauwer. "A Tensor-Based Method for Large-Scale Blind Source Separation Using Segmentation". In: *IEEE Transactions on Signal Processing* 65.2 (Jan. 2017), pp. 346–358. ISSN: 1941-0476. DOI: 10.1109/TSP.2016.2617858.

[12] Yanzhao Cao, Somak Das, and Hans-Werner van Wyk. "Adaptive Gradient Descentfor Optimal Control of Parabolic Equations with Random Parameters". 2021.

[13] Yanzhao Cao et al. "Analysis of the Stochastic Alternating Least Squares Method for the Decomposition of Random Tensors". 2021.

[14] J. Douglas Carroll and Jih-Jie Chang. "Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition". In: *Psychometrika* 35.3 (Sept. 1970), pp. 283–319. DOI: 10.1007/bf02310791.

[15] Dehua Cheng et al. "SPALS: Fast Alternating Least Squares via Implicit Leverage Scores Sampling". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 721–729. ISBN: 9781510838819.

[16] F. Chollet. *Deep Learning with Python*. Manning Publications, 2017. ISBN: 9781638352044. URL: https://books.google.com/books?id=wzozEAAAQBAJ.

[17] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *J. Mach. Learn. Res.* 12.null (July 2011), pp. 2121–2159. ISSN: 1532-4435.

[18] Nicholas Eriksson. "Algebraic Statistics for computational biology". In: vol. 13. Cambridge University Press, 2005. Chap. Tree construction using singular value decomposition, pp. 347–358.

[19] Lawrence Evans. *Partial differential equations*. Providence, R.I: American Mathematical Society, 2010. ISBN: 9780821849743.

[20] Nicolaas (Klaas) M. Faber, Rasmus Bro, and Philip K. Hopke. "Recent developments in CANDECOMP/PARAFAC algorithms: a critical review". In: *Chemometrics and Intelligent Laboratory Systems* 65.1 (Jan. 2003), pp. 119–137. DOI: 10.1016/s0169-7439(02)00089-8.

[21] Caroline Geiersbach and Georg Ch. Pflug. "Projected Stochastic Gradients for Convex Constrained Problems in Hilbert Spaces". In: *SIAM Journal on Optimization* 29.3 (Jan. 2019), pp. 2079–2099. DOI: 10.1137/18m1200208.

[22] C. J. Gittelson. "Stochastic Galerkin Discretization of the Log-Normal Isotropic Diffusion Problem". In: *Mathematical Models and Methods in Applied Sciences* 20.02 (Feb. 2010), pp. 237–263. DOI: 10.1142/s0218202510004210.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[24] Luigi Grippo and Marco Sciandrone. "Globally convergent block-coordinate techniques for unconstrained optimization". In: *Optimization Methods and Software* 10.4 (Jan. 1999), pp. 587–637. DOI: 10.1080/10556789908805730.

[25] Guifang Guo et al. “Three-dimensional thermal finite element modeling of lithium-ion battery in thermal abuse application”. In: *Journal of Power Sources* 195.8 (Apr. 2010), pp. 2393–2398. DOI: 10.1016/j.jpowsour.2009.10.090.

[26] C. J. Hillar and L.-H. Lim. “Most tensor problems are NP hard”. In: *J. ACM* 60.6 (2013).

[27] Einar Hille and Ralph S. Phillips. *Functional analysis and semi-groups*. Third printing of the revised edition of 1957, American Mathematical Society Colloquium Publications, Vol. XXXI. American Mathematical Society, Providence, R. I., 1974, pp. xii+808.

[28] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent”. In: *Cited on* 14.8 (2012).

[29] Mariya Ishteva, Haesun Park, and Le Song. “Unfolding latent tree structures using 4th order tensors”. In: *International Conference on Machine Learning*. 2013, pp. 316–324.

[30] Jichun. *Computational partial differential equations using MATLAB*. Boca Raton: CRC Press, 2009. ISBN: 978-1420089042.

[31] C. G. Khatri and C. Radhakrishna Rao. “Solutions to Some Functional Equations and Their Applications to Characterization of Probability Distributions”. In: *Sankhya: The Indian Journal of Statistics, Series A (1961-2002)* 30.2 (1968), pp. 167–180. ISSN: 0581572X. URL: http://www.jstor.org/stable/25049527.

[32] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (Dec. 22, 2014). arXiv: 1412.6980v9 [cs.LG].

[33] Tamara G. Kolda and Brett W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500. DOI: 10.1137/07070111x.

[34] Tamara G. Kolda and David Hong. "Stochastic Gradients for Large-Scale Tensor Decomposition". In: *SIAM Journal on Mathematics of Data Science* 2.4 (Jan. 2020), pp. 1066–1095. DOI: 10.1137/19M1266265. arXiv: 1906.01687v1 [math.NA].

[35] Tamara Kolda et al. *Tensor Toolbox for MATLAB v. 3.0*. Mar. 2017. DOI: 10.11578/dc.20201001.24. URL: https://www.osti.gov//servlets/purl/1349514.

[36] Na Li, Stefan Kindermann, and Carmeliza Navasca. "Some convergence results on the Regularized Alternating Least-Squares method for tensor decomposition". In: *Linear Algebra and its Applications* 438.2 (Jan. 2013), pp. 796–812. DOI: 10.1016/j.laa.2011.12.002.

[37] Xiaoyu Li and Francesco Orabona. "On the Convergence of Stochastic Gradient Descent with Adaptive Stepsizes". In: ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, Apr. 2019, pp. 983–992. URL: http://proceedings.mlr.press/v89/li19c.html.

[38] Lions. *Optimal control of systems governed by partial differential equations*. Berlin,New York: Springer-Verlag, 1971. ISBN: 9780387051154.

[39] Takanori Maehara, Kohei Hayashi, and Ken-ichi Kawarabayashi. "Expected Tensor Decomposition with Stochastic Gradient Descent". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 1919–1925. URL: http://dl.acm.org/citation.cfm?id=3016100.3016167.

[40] Julien Mairal. "Stochastic Majorization-Minimization Algorithms for Large-Scale Optimization". In: *NIPS 2013 - Advances in Neural Information Processing Systems*. Ed. by C.J.C. Burges et al. Advances in Neural Information Processing Systems 26. South Lake Tahoe, United States, Dec. 2013, pp. 2283–2291. URL: https://hal.inria.fr/hal-00835840.

[41] Matthieu Martin, Sebastian Krumscheid, and Fabio Nobile. *Analysis of stochastic gradient methods for PDE-constrained optimal control problems with uncertain parameters*. Tech. rep. EPFL, 2018.

[42] Pamela McCorduck. *Machines who think : a personal inquiry into the history and prospects of artificial intelligence*. Natick, Mass: A.K. Peters, 2004. ISBN: 978-1568812052.

[43] H Brendan McMahan and Matthew Streeter. "Adaptive Bound Optimization for Online Convex Optimization". In: *COLT 2010* (2010), p. 244.

[44] V. A. Morozov. *Methods for Solving Incorrectly Posed Problems*. Springer New York, 1984. DOI: 10.1007/978-1-4612-5280-1.

[45] C. Navasca, L. De Lathauwer, and S. Kindermann. "Swamp reducing technique for tensor decomposition". In: *2008 16th European Signal Processing Conference*. 2008, pp. 1–5.

[46] A. Nemirovski et al. "Robust Stochastic Approximation Approach to Stochastic Programming". In: *SIAM Journal on Optimization* 19.4 (Jan. 2009), pp. 1574–1609. DOI: 10.1137/070704277.

[47] Yu. Nesterov. "Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems". In: *SIAM Journal on Optimization* 22.2 (Jan. 2012), pp. 341–362. DOI: 10.1137/100802001.

[48] Monty Newborn. *Kasparov versus Deep Blue : Computer Chess Comes of Age*. New York, NY: Springer New York, 1997. ISBN: 978-1-4612-2260-6.

[49] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, Jan. 2000. DOI: 10.1137/1.9780898719468.

[50] Gerard Regan. *A brief history of computing*. London: Springer, 2012. ISBN: 978-1447123583.

[51] Robert R. Richardson, Shi Zhao, and David A. Howey. "On-board monitoring of 2-D spatially-resolved temperatures in cylindrical lithium-ion batteries: Part I. Low-order thermal modelling". In: *Journal of Power Sources* 326 (Sept. 2016), pp. 377–388. DOI: 10.1016/j.jpowsour.2016.06.103.

[52] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (Sept. 1951), pp. 400–407. DOI: 10.1214/aoms/1177729586.

[53] R.Tyrrell Rockafellar and Stanislav Uryasev. "Conditional value-at-risk for general loss distributions". In: *Journal of Banking & Finance* 26.7 (July 2002), pp. 1443–1471. DOI: 10.1016/s0378-4266(02)00271-6.

[54] N.D. Sidiropoulos, G.B. Giannakis, and R. Bro. "Blind PARAFAC Receivers for DS-CDMA Systems". In: *IEEE Trans. on Signal Processing* 48.3 (2000), pp. 810–823.

[55] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-Way Analysis with Applications in the Chemical Sciences*. John Wiley & Sons, Ltd, Aug. 2004. DOI: 10.1002/0470012110.

[56] André Uschmajew. "Local Convergence of the Alternating Least Squares Algorithm for Canonical Tensor Approximation". In: *SIAM Journal on Matrix Analysis and Applications* 33.2 (Jan. 2012), pp. 639–652. DOI: 10.1137/110843587.

[57] Nico Vervliet and Lieven De Lathauwer. "A Randomized Block Sampling Approach to Canonical Polyadic Decomposition of Large-Scale Tensors". In: *IEEE Journal of Selected Topics in Signal Processing* 10.2 (Mar. 2016), pp. 284–295. DOI: 10.1109/jstsp.2015.2503260. URL: https://doi.org/10.1109/jstsp.2015.2503260.

[58] Liqi Wang and Moody T. Chu. "On the Global Convergence of the Alternating Least Squares Method for Rank-One Approximation to Generic Tensors". In: *SIAM*

Journal on Matrix Analysis and Applications 35.3 (Jan. 2014), pp. 1058–1072. DOI: 10.1137/130938207.

[59] Z. Wang, J. Ma, and L. Zhang. "Finite Element Thermal Model and Simulation for a Cylindrical Li-Ion Battery". In: *IEEE Access* 5 (2017), pp. 15372–15379.

[60] Rachel Ward, Xiaoxia Wu, and Léon Bottou. "AdaGrad Stepsizes: Sharp Convergence Over Nonconvex Landscapes". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, June 2019, pp. 6677–6686. URL: http://proceedings.mlr.press/v97/ward19a.html.

[61] Michael Wooldridge. *A brief history of artificial intelligence : what it is, where we are, and where we are going*. New York: Flatiron Books, 2021. ISBN: 978-1250770745.

[62] B. Xu et al. "Modeling of Lithium-Ion Battery Degradation for Cell Life Assessment". In: *IEEE Transactions on Smart Grid* 9.2 (2018), pp. 1131–1140.

[63] Yangyang Xu and Wotao Yin. "A Globally Convergent Algorithm for Nonconvex Optimization Based on Block Coordinate Update". In: *Journal of Scientific Computing* 72 (2017), pp. 700–734. ISSN: 0885-7474. DOI: 10.1007/s10915-017-0376-0.

[64] Yangyang Xu and Wotao Yin. "Block Stochastic Gradient Iteration for Convex and Nonconvex Optimization". In: *SIAM Journal on Optimization* 25.3 (Jan. 2015), pp. 1686–1716. DOI: 10.1137/140983938.