ABSTRACT

Title of Dissertation: OPTIMIZATION UNDER STOCHASTIC ENVIRONMENT

Yunchuan Li
Doctor of Philosophy, 2020

Dissertation Directed by: Professor Michael Fu
Department of Electrical and Computer Engineering

Stochastic optimization (SO) is extensively studied in various fields, such as control engineering, operations research, and computer science. It has found wide applications ranging from path planning (civil engineering) and tool-life testing (industrial engineering) to Go-playing artificial intelligence (computer science). However, SO is usually a hard problem primarily because of the added complexity from random variables. The objective of this research is to investigate three types of SO problems: single-stage SO, multi-stage SO and fast real-time parameter estimation under stochastic environment.

We first study the single-stage optimization problem. We propose Direct Gradient Augmented Response Surface Methodology (DiGARSM), a new sequential first-order method for optimizing a stochastic function. In this approach, gradients of the objective function with respect to the desired parameters are utilized in addition to response measurements. We intend to establish convergence of the proposed method, as well as traditional approaches which do not use gradients. We expect an improvement in convergence speed with the added derivative information.

Second, we analyze a tree search problem with an underlying Markov decision

process. Unlike traditional tree search algorithms where the goal is to maximize the cumulative reward in the learning process, the proposed method aims at identifying the best action at the root that achieves the highest reward. A new tree algorithm based on ranking and selection is proposed. The selection policy at each node aims at maximizing the probability of correctly selecting the best action.

The third topic is motivated by problems arising in neuroscience, specifically, a Maximum Likelihood (ML) parameter estimation of linear models with noise-corrupted observations. We developed an optimization algorithm designed for non-convex, linear state-space model parameter estimation. The ML estimation is carried out by the Expectation-Maximization algorithm, which iteratively updates parameter estimates based on the previous estimates. Since the likelihood surface is in general non-convex, a model-based global optimization method called Model Reference Adaptive Search (MRAS) is applied.

OPTIMIZATION UNDER STOCHASTIC ENVIRONMENT

by

Yunchuan Li

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Michael Fu, Chair/Advisor
Professor Steven Marcus
Professor Behtash Babadi
Professor P. S. Krishnaprasad
Professor Thomas Goldstein

Dedication

To my family.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First, I would like to thank my advisor, Professor Michael Fu for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past five years. He has always made himself available for help and advice. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank Professor Steven Marcus and Professor Behtash Babadi for leading in-depth discussions and offering invaluable guidance, without which this thesis would have been a distant dream. Thanks are due to Professor P. S. Krishnaprasad, Nuno Martins, Michael Rotkowitz, Ilya Ryzhov, and Rama Chellappa for excellent classes an dedicated teaching.

This journey would not be possible without the support of my friends and fellow students at the University of Maryland. To my colleagues in the research group, Proloy Das, James Ferlez, Cheng Jie, Joshua Kulasingham, Sina Miran, Bhaskar Ramasubramanian and Guowei Sun, thank you for interesting discussions across many topics. To Sheng Cheng, David Hartman, Zhenyu Lin, Manasij Venkatesh, Daiwei Zhu, the conversations, game nights, rides, general helps and friendship are greatly appreciated. I am especially grateful to Miss Xiaoxu Meng, for supporting me in all of my pursuits and encouraging me to follow my dreams, both academically and emotionally.

I owe my deepest thanks to my family - my mother and father who have always stood by me and guided me through my career. Thank you to my mother, for supporting

the family during the hard times. I would also like to thank doctors and medical workers at West China Hospital, especially Dr. Jiayin Yang, Dr. Tao Lyu and anonymous organ donors, who saved my and many others' family.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The problem of optimization in a stochastic environment arises from various communities and applications, such as computer science, engineering and finance. Randomness comes into play in a number of ways. It affects the cost function to be optimized, for instance the response of the objective function to be optimized. On the other hand, randomness influences dynamics of the target systems, as in Markov decision problems. Historically, decision making under uncertainty is a hard problem because of the added complexity from randomness.

To formulate the problem, suppose $\mathscr{X} \subset R^d$ is the domain of variable x. The problem of interest is to minimize a scalar-valued function of x, $f(x)$, which represents the expected loss incurred by exercising action x (or maximize $f(x)$, which would represent the gain/profit by x). However, there are several challenges. First, the analytical form of $f(x)$ may be unknown, so the search for an optimum can only be done by examining sample measurements of $f(x)$. Second, the measurements may be expensive, which prevents exhausting all possibilities and finding the optimum by brute force. Third, in stochastic settings, the measurement of the objective function is corrupted by noise. Take the following inventory control problem as an illustration.

Example 1.1. *Suppose that a company keeps a warehouse. On each day, it will decide*

an order quantity of a product (x). Many factors contribute to the operational cost of the company, for example, holding cost for remaining product, and a penalty for unsatisfied demand. The objective is to minimize the total cost. However, the demand each day is unknown and random. Even if the same order amount is exercised everyday, the outcomes are different. In addition, each sample takes one day to collect.

Therefore, the goal is to find a cost-efficient algorithm to solve

$$\min_{x \in \mathscr{X}} \{f(x) = \mathbb{E}[\tilde{f}(x)]\}. \tag{1.1}$$

In this example, $\tilde{f}(x)$ represents the (noisy) sum of holding cost and penalty cost.

Remark 1.1. *Due to the large variations in the field of stochastic optimization, different notation systems will be employed depending on the problem setting. For example, I use "x" for the decision variable in single-stage optimization problem, and "a" as the decision in another problem, i.e., the notation that is most familiar to each community is applied.*

Perhaps the most evident distinction in stochastic optimization is between the solution strategies for single-stage (state-independent) problems and those for multi-stage (state-dependent) problems. The single-stage problems aim at finding a single optimal value of the objective function or identifying the best parameters that achieve this optimum. Examples include optimizing a deterministic function with noisy samples. On the other hand, multi-stage problems involve a sequence of decisions, and after each decision is made, the underlying state changes. The dynamics of the state transition may be unknown and stochastic, and will have impact on the objective function. The inventory

control problem in Example 1.1 is an illustration of such problems.

In my research, I address both types of stochastic optimization problems: single stage and multi-stage stochastic optimization. In addition, parameter identification with noisy observations is addressed in this work.

The rest of this dissertation is organized as follows: In Chapter 2, we proposes a stochastic approximation algorithm, DiGARSM, which aims at optimizing an objective function with noisy observations [1, 2]. Chapter 3 investigates a tree policy for a Monte Carlo tree search algorithm, where the objective is locating the best action for a tree search problem [3, 4]. Finally, we investigate a parameter estimation problem in Chapter 4. An expectation-maximization algorithm is proposed, and a model-based global optimization algorithm is applied to solve this nonconvex problem [5].

Chapter 2: Single-Stage Stochastic Optimization

2.1 Background

Single-stage stochastic optimization tries to optimize a (random) objective function subject to (random) constraints. Formally, consider the unconstrained problem

$$\min_{x \in R^d} f(x) = \min_{x \in R^d} \mathbb{E}[\tilde{f}(x)], \tag{2.1}$$

where $\tilde{f}(x)$ is a stochastic function with mean $f(x)$. Besides the response samples, a number of techniques to estimate the gradient of performance measure through samples have been proposed. Examples of such methods include perturbation analysis [6, 7] and the likelihood ratio method [8]. These gradient estimates have been applied extensively to stochastic approximation [9]. If the stochastic gradient estimate of the objective function is available, derivative-based stochastic approximation can thus be applied to solve these kind of problems iteratively, i.e., at iteration k the point to be explored is x_k, the next iterate is found by

$$x_{k+1} = x_k - a_k g_k, \tag{2.2}$$

where a_k is the step size, and g_k represents the search direction.

The first such algorithm was introduced by Robbins and Monro [10] with some mild constraints on the learning rate. Following that, numerous developments of stochastic approximation emerge. For instance, [11] extends to multidimesional problems and further proves almost sure convergence of the algorithm. More recently, many variations of stochastic gradient descent have been investigated. For example, AdaGrad [12] adapts the learning rate to parameters to be optimized, and Adam [13], in addition, uses the exponentially decaying average of squared past gradients.

However, many stochastic approximation algorithms assume an unpolluted measurement of stochastic gradient is available, which is not always true. For example, the analytical form of $f(x)$ may not be available in many engineering practice, and only noisy samples can be obtained. In these cases, perhaps the most intuitive solution is to calculate the numerical gradient based on the noisy samples $\tilde{f}(x)$ [14]. Other methods include using the samples to fit a local metamodel, and deriving the next search direction according to the fitted model. For example, Response Surface Methodology (RSM) uses the samples to fit a linear model, and the gradient of the fitted linear model works as the search direction.

In some settings, both (noisy) function and its gradient samples are available. Therefore, one way to improve the existing derivative-based algorithms is to utilize the two types of samples. Several methods were proposed addressing this problem to varying degrees. To illustrate, [15–17] utilize both gradient and response measurements with proven record of improvement. Nonetheless, to the best of our knowledge, there is little literature that investigates the combining gradient and response samples with local metamodel

methods. Therefore, it is worthy to focus on this line of research.

One of the most popular metamodel methods in simulation optimization is the aforementioned RSM, which was first described in [18] and was applied to a real physical system. Since that time, the use of RSM has been extended successfully to many other scientific fields such as biometrics [19], industrial engineering [20] and materials science [21]. RSM is often applied in optimizing stochastic simulation models. One of the earliest case studies is given in [22]. Other examples of RSM in simulation include [9] and [23]. More recent developments of RSM in simulation are discussed in [24–26]. [27] gives a review of developments in RSM in the period 1966-1988. More recent overviews of RSM can be found in [28].

Traditionally, RSM views the system to be optimized as a black box and is able to obtain the input-output pairs (variable-response pairs) from the model. It uses a sequence of local experiments that leads to the optimum. In each local experiment, a number of input-output pairs are observed in a small region. A metamodel, which is usually a first or second-order polynomial model, is then used to fit the response surface. Steepest descent (or ascent) is performed to determine the next region to be explored, where the search direction is given by the fitted model. The fit and search process is repeated until a satisfactory result has been obtained; see [28] for details. To determine input points to measure in each local experiment, several design methods are presented, e.g., factorial design, Plackett–Burman design [29] and simplex design. More complex design methods include robust parameter design; see [30] for details. A successful design should be examined based on several criteria, such as prediction variance [31]. Experiment design and optimization method for multiple-response problems have also been studied. For

example, [32] considers designs for systems with two correlated responses, and more general multiple-response systems are studied in [33].

Though RSM is only a heuristic [28], it works well in applications when relatively accurate response measurements are available. However, the measurement on the output of the system is often noisy, which could lead to unstable behavior of RSM. In such situation, additional stochastic gradient information can be applied to improve the performance of RSM. One big question here is how to combine the gradient samples into RSM.

With additional gradient information, a modified regression model-Direct Gradient Augmented Regression (DiGAR)-is investigated in [34]. DiGAR fits a regression model using both response and gradient information with a least squares approach. This regression model shows great potential in the presence of significant response measurement noise. Under some mild assumptions, it is also shown that the estimator of the gradient is unbiased. Therefore, we expect the modified RSM with DiGAR model will perform better than traditional RSM with regular least-squares regression model. Moreover, since gradient augmented RSM uses both response and gradient measurements, we also believe that in cases where gradient information is unreliable but response measurement is accurate, i.e., high variance in gradient measurement but low in response measurement, the modified RSM should still perform well.

With both gradient and response samples available, we propose an iterative local metamodel method called *Direct Gradient Augmented Response Surface Methodology* (DiGARSM) to solve Problem Equation (2.1). DiGARSM was first proposed in [1]. However, from the practical viewpoint, there are three challenges remain to be addressed.

Heuristically, to find a better search direction, the more reliable information should be utilized. For instance, if the gradient measurements are noisier (e.g., have higher variances) than the response measurements, the DiGAR process should rely more on the responses. Generally, the uncertainty of response and gradient measurements is unknown a priori and can only be inferred from the samples. Therefore, the first challenge is to design a procedure that balances the uncertainties in the measurements.

Second, the original DiGARSM in [1] applies a full factorial design. Though this provides a more accurate regression model and thus a better search direction, its computational and measurement cost is exponential in the number of dimensions, i.e., 2^d where d denotes the number of dimensions, which becomes prohibitively expensive when d is large. Thus, our second challenge is to make the approach scaleable to high dimensions.

Another drawback in applying RSM is the lack of theoretical convergence guarantee. Some prior work related to the convergence property of RSM includes stopping rules [35] and confidence regions [36]. Theoretical performance of RSM incorporated with a trust region method is presented by [37]. However, to the best of our knowledge, there is little research on the convergence analysis of RSM. Thus, our third challenge is to rigorously establish that RSM, including the version augmented with direct gradient information, converges to the optimum and investigate the convergence rate.

To address the first challenge, we propose to add weighting parameters on the loss of the regression process. As a result, the final search direction will be composed of weighted response and gradient measurements. Sample variance estimates will be used for weight tuning. To address the second and third challenges, we propose to analyze DiGARSM by framing it as a stochastic approximation algorithm and incorporating the simultaneous

perturbation (SP) method [38] in the experiment design stage to lower the computational and measurement cost. Under some mild assumptions, convergence analysis methods (e.g., [14, 39]) can be applied to DIGARSM.

2.2 Problem Formulation

2.2.1 Stochastic Approximation with RSM

SA generates a sequence of iterates $\{x_k\}$ using the recursion Equation (2.2). Central to such algorithms is the estimation of gradient direction g_k, and RSM provides one method to estimate g_k with appropriate designs: input-output samples around iterate x_k are taken, and used to fit a linear model. Then the search direction g_k is provided by the fitted linear model. Suppose that at iteration k we sample symmetrically in each dimension of x_k with a full-factorial design, i.e., sample $x_{k,j} := x_k + c_k\theta_j$, where c_k is a positive sequence and $\theta_j \in \{[\pm t_1, \pm t_2, \ldots, \pm t_d]^T\}$ is the perturbation vector and the l-th component takes value either $+t_l$ or $-t_l$. For simplicity, let $\mathscr{S}^d$ be the set of indices for θ_j, i.e., θ_j takes all values in $\{[\pm t_1, \pm t_2, \ldots, \pm t_d]^T\}$ for $j \in \mathscr{S}^d$. Further assume that each point is sampled n times (in total, $n2^d$ samples). The i-th response and gradient sample at x are denoted by $\tilde{f}(x, \omega_i)$ and $\widetilde{\nabla f}(x, \omega_i)$, respectively. Denote the optimal point by

$$x^* = \arg\min_x f(x).$$

Traditional RSM would use the n sets of noisy response samples to fit a linear model

of the form

$$\hat{f}(x) = \beta_{k0} + \beta_k^T x,$$

where $\beta_k = [\beta_{k1}, \ldots, \beta_{kd}]^T$. The loss function is given by the sum of squared errors

$$\begin{aligned} L &= \sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} (\tilde{f}(x_{k,j}, \omega_i) - \hat{f}(x_{k,j}))^2 \\ &= \sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} (\tilde{f}(x_{k,j}, \omega_i) - \beta_{k0} - \beta_k^T x_{k,j})^2, \end{aligned}$$

where $\tilde{f}(x_{k,j}, \omega_i)$ denotes the i-th response sample taken at $x_{k,j}$ for $i = 1, 2, \ldots, n$. The optimal parameters $\hat{\beta}_{k0}$ and $\hat{\beta}_k$ are found by minimizing the loss function. Taking the derivative and setting to 0 yields the optimal $\hat{\beta}_k$:

$$\hat{\beta}_k = [\sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} (x_{k,j} - \bar{x}_k)(x_{k,j} - \bar{x}_k)^T]^{-1} [\sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} (x_{k,j} - \bar{x}_k)(\tilde{f}(x_{k,j}, \omega_i) - \bar{\bar{f}}_k)], \tag{2.3}$$

where $\bar{x}_k$ and $\bar{\bar{f}}_k$ are the means over all sampled points and sample responses at iteration k, respectively. The gradient estimate is then given by $\hat{\beta}_k$, i.e., $g_k = \hat{\beta}_k$.

Remark 2.1. *When candidate points are sampled symmetrically (e.g., full factorial design in this work), $\bar{x}_k = x_k$, the sample response mean is defined by*

$$\bar{\bar{f}}_k = \frac{1}{n2^d} \sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} \tilde{f}(x_{k,j}, \omega_i). \tag{2.4}$$

2.2.2 Stochastic Approximation with Direct Gradient Augmented RSM

Now we present DiGARSM, in which we assume both response measurements and direct gradient estimates are available at the time of sampling, i.e., we can acquire both $\tilde{f}(x_{k,j},\omega_i)$ and $\widetilde{\nabla f}(x_{k,j},\omega_i)$ when we sample point $x_{k,j}$. With the additional gradient information, we slightly modify the linear model to be fit in order to incorporate the gradient. Following [34], we fit the response and gradient samples to

$$\hat{f}(x) = \beta_{k0} + \beta_k^T x, \tag{2.5}$$

$$\widehat{\nabla f}(x) = \beta_k = [\beta_{k1}, \beta_{k2}, \ldots, \beta_{kd}]^T. \tag{2.6}$$

Since there is an additional fitting term, the loss function is therefore revised accordingly:

$$\begin{aligned} L =& \alpha_0 \sum_{j\in\mathscr{S}^d}\sum_{i=1}^{n}(\tilde{f}(x_{k,j},\omega_i) - \hat{f}(x_{k,j}))^2 + \\ & \sum_{j\in\mathscr{S}^d}\sum_{i=1}^{n}(\widetilde{\nabla f}(x_{k,j},\omega_i) - \widehat{\nabla f}(x_{k,j}))^T W (\widetilde{\nabla f}(x_{k,j},\omega_i) - \widehat{\nabla f}(x_{k,j})) \\ =& \alpha_0 \sum_{j\in\mathscr{S}^d}\sum_{i=1}^{n}(\tilde{f}(x_{k,j},\omega_i) - \beta_{k0} - \beta_k^T x_{k,j})^2 + \\ & \sum_{j\in\mathscr{S}^d}\sum_{i=1}^{n}(\widetilde{\nabla f}(x_{k,j},\omega_i) - \beta_k)^T W (\widetilde{\nabla f}(x_{k,j},\omega_i) - \beta_k), \end{aligned} \tag{2.7}$$

where α_0 is a weight parameter and W is a diagonal weight matrix $W = diag([\alpha_1, \alpha_2, \ldots, \alpha_d])$, with $\sum_{i=0}^{d}\alpha_i = 1$ and $\alpha_i \geq 0\ \forall i = 0, 1, \ldots, d$. The weights are intended to balance between each dimension of gradient estimates and responses.

The additional term $\sum_{j\in\mathscr{S}^d}\sum_{i=1}^{n}(\widetilde{\nabla f}(x_{k,j},\omega_i) - \beta_k)^T W (\widetilde{\nabla f}(x_{k,j},\omega_i) - \beta_k)$ represents

the weighted squared error between the gradient samples and the fitted model, which restricts the derivative of the fitted linear model to be close to the noisy observations.

The minimizing β_k (hence g_k) is given by

$$\hat{\beta}_k = [\alpha_0 \sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} (x_{k,j} - \bar{x}_k)(x_{k,j} - \bar{x}_k)^T + 2^d n W]^{-1} \\ [\alpha_0 \sum_{j \in \mathscr{S}^d} \sum_{i=1}^{n} (x_{k,j} - \bar{x}_k)(\tilde{f}(x_{k,j}, \omega_i) - \bar{\tilde{f}}_k) + 2^d n W \widetilde{\nabla f_k}], \tag{2.8}$$

where $\widetilde{\nabla f_k}$ is the sample mean of gradient measurements defined analogously to Equation (2.4).

Note that if α_0 is set to 0, only gradient information is used, and DiGARSM becomes a Robbins-Monro (RM) stochastic approximation [10]. When α_0 is set to 1 (W is a matrix of 0's), only response samples are utilized, and DiGARSM reduces to regular RSM as in Equation (2.3), which will be referred to as the "Standard" form and will be evaluated in Section 2.6.

The algorithmic description of RSM and DiGARSM stochastic approximation is given in Algorithm 1.

2.3 Weight Tuning in DiGARSM

One question that follows naturally after introducing weight parameters is how to set them in practice. In this section, we present the optimal choices of weights for DiGARSM, where the optimality is evaluated by the variance of the gradient estimator.

Proposition 2.1 (DiGARSM variance minimization)**.** *For the gradient estimator provided*

Algorithm 1: (DiGA)RSM Stochastic Approximation.

Input: Initial point x_0, weights $\alpha_0, \ldots, \alpha_d$ (for DiGARSM), positive sequences $\{a_k\}$ and $\{c_k\}$, perturbations $\{t_l\}_{l=1}^{d}$, number of samples/replications for each point n

Output: Optimal point $x^* = \arg\min_x f(x)$

$k \leftarrow 0$
while *stopping rule not met* **do**
  for $i = 1, 2, \ldots, n$ *and* $j = 1, 2, \ldots, 2^d$ **do**
    Calculate the set of points to be sampled:
    $x_{k,j} = x_k + c_k \theta_j$, $\theta_j \in \{[\pm t_1, \pm t_2, \ldots, \pm t_d]^T\}$
    Obtain samples of response and gradient:
    $\tilde{f}(x_{k,j}, \omega_i)$ and $\widetilde{\nabla f}(x_{k,j}, \omega_i)$
  end
  Calculate gradient estimate g_k with Equation (2.3) (RSM) or Equation (2.8) (DiGARSM)
  $x_{k+1} \leftarrow x_k - a_k g_k$
  $k \leftarrow k + 1$
end
return x_k

by DiGARSM in Equation (2.8), assume homogeneous variance and independence between different sample points, i.e.,

1. $\mathrm{Var}(\tilde{f}(x_{k,j}, \omega_i)) = \sigma_f^2$ *for all* $j = 1, 2, \ldots, 2^n$ *and* $i = 1, 2, \ldots, n$,

2. $\mathrm{Var}(\widetilde{\nabla_l f}(x_{k,j}, \omega_i)) = \sigma_{g,l}^2$ *for all* $j = 1, 2, \ldots, 2^n$ *and* $i = 1, 2, \ldots, n$, *where* $\widetilde{\nabla_l f}(x_{k,j}, \omega_i)$ *is the* l*-th gradient component,*

3. $$Cov(\tilde{f}(x_{k,j}, \omega_i), \widetilde{\nabla_l f}(x_{k,m}, \omega_n)) = \begin{cases} \sigma_{f/g,l}^2, \ if \ i = n \ and \ j = m \\ 0, \ otherwise \end{cases},$$

   and

4. $\tilde{f}(x_{k,j}, \omega_i) \perp \tilde{f}(x_{k,m}, \omega_n)$ *and* $\widetilde{\nabla_l f}(x_{k,j}, \omega_i) \perp \widetilde{\nabla_l f}(x_{k,m}, \omega_n)$ *for all* $i \neq n$ *or* $j \neq m$,

then the variance of the l-th element of the gradient estimator is minimized if the weights are inversely proportional to the variances:

$$\frac{\alpha_0^*}{\alpha_l^*} = \frac{\sigma_{g,l}^2}{\sigma_f^2},\ l = 1,2,\ldots,d, \tag{2.9}$$

i.e.,

$$\alpha_0^* = \frac{1}{1+\sum_{m=1}^{d}\sigma_f^2/\sigma_{g,m}^2}, \tag{2.10}$$

$$\alpha_l^* = \frac{\sigma_f^2}{\sigma_{g,l}^2}\cdot\frac{1}{1+\sum_{m=1}^{d}\sigma_f^2/\sigma_{g,m}^2},\ = 1,2,\ldots,d. \tag{2.11}$$

Proof. Under DiGARSM settings, the gradient estimator is given by Equation (2.8). Note that

$$\sum_{j=1}^{2^d}\sum_{i=1}^{n}(c_k\theta_j)(c_k\theta_j)^T = \alpha_0 n 2^d c_k^2\Theta,$$

where $\Theta = diag([t_1^2, t_2^2, \ldots, t_d^2])$. Therefore,

$$\begin{aligned}\hat{\beta}_k =& [\alpha_0\sum_{j=0}^{2^d}\sum_{i=1}^{n}(c_k\theta_j)(c_k\theta_j)^T + 2^d nW]^{-1}[\alpha_0\sum_{j=1}^{2^d}\sum_{i=1}^{n}(c_k\theta_j)(\tilde{f}(x_k+c_k\theta_j,\omega_i)-\bar{\tilde{f}}_k) + 2^d nW\widetilde{\nabla f_k}]\\ =& \frac{1}{n2^d}N_k[\alpha_0\sum_{j=1}^{2^d}\sum_{i=1}^{n}(c_k\theta_j)(\tilde{f}(x_k+c_k\theta_j,\omega_i)-\bar{\tilde{f}}_k) + 2^d nW\widetilde{\nabla f_k}],\end{aligned}$$

where

$$\begin{aligned} N_k =& [\alpha_0 c_k^2 \Theta + W]^{-1} \\ =& diag((\alpha_0 c_k^2 t_1^2 + \alpha_1)^{-1}, (\alpha_0 c_k^2 t_2^2 + \alpha_2)^{-1}, \ldots, (\alpha_0 c_k^2 t_d^2 + \alpha_d)^{-1}). \end{aligned}$$

The l-th component of $\hat{\beta}_k$ can be expressed by

$$\begin{aligned} \hat{\beta}_{k,l} =& e_l^T \hat{\beta}_k \\ =& \frac{1}{n2^d} e_l^T N_k [\alpha_0 \sum_{j=1}^{2^d} \sum_{i=1}^{n} (c_k \theta_j)(\tilde{f}(x_k + c_k \theta_j, \omega_i) - \bar{\bar{\tilde{f}}}_k) + 2^d n W \widetilde{\overline{\nabla f_k}}] \\ =& \frac{1}{n2^d (\alpha_0 c_k^2 t_l^2 + \alpha_l)} e_l^T [\alpha_0 \sum_{j=1}^{2^d} \sum_{i=1}^{n} (c_k \theta_j \tilde{f}(x_k + c_k \theta_j, \omega_i)) + 2^d n W \widetilde{\overline{\nabla f_k}}] \end{aligned}$$

the term associated with $\bar{\bar{\tilde{f}}}_k$ vanishes due to the symmetric property of θ_j, i.e., $\sum_{j=1}^{2^d} \theta_j = 0$,

so

$$\begin{aligned} \hat{\beta}_{k,l} =& \frac{1}{n2^d (\alpha_0 c_k^2 t_l^2 + \alpha_l)} [\alpha_0 \sum_{j=1}^{2^d} \sum_{i=1}^{n} (c_k t_l \tilde{f}(x_k + c_k \theta_j, \omega_i)) + 2^d n(\alpha_l) e_l^T \widetilde{\overline{\nabla f_k}}] \\ =& \frac{\alpha_0}{n2^d (\alpha_0 c_k^2 t_l^2 + \alpha_l)} \sum_{j=1}^{2^d} \sum_{i=1}^{n} [c_k t_l \tilde{f}(x_k + c_k \theta_j, \omega_i) + \frac{\alpha_l}{\alpha_0} \nabla_l \tilde{f}(x_k + c_k \theta_j, \omega_i)]. \end{aligned}$$

By homogeneity, the variance of the l-th element of the gradient estimator is given by

$$\begin{aligned}
\mathrm{Var}(\hat{\beta}_{k,l}) =& [\frac{\alpha_0}{n2^d(\alpha_0 c_k^2 t_l^2 + \alpha_l)}]^2 \sum_{j=1}^{2^d}\sum_{i=1}^{n} \mathrm{Var}(c_k t_l \tilde{f}(x_k + c_k\theta_j, \omega_i) + \frac{\alpha_l}{\alpha_0}\nabla_l \tilde{f}(x_k + c_k\theta_j, \omega_i)) \\
=& [\frac{\alpha_0}{n2^d(\alpha_0 c_k^2 t_l^2 + \alpha_l)}]^2 \sum_{j=1}^{2^d}\sum_{i=1}^{n} [c_k^2 t_l^2 \mathrm{Var}(\tilde{f}(x_k + c_k\theta_j, \omega_i)) + (\frac{\alpha_l}{\alpha_0})^2 \mathrm{Var}(\nabla_l \tilde{f}(x_k + c_k\theta_j, \omega_i)) + \\
& 2c_k\theta_{j,l}\frac{\alpha_l}{\alpha_0}\mathrm{Cov}(\tilde{f}(x_k + c_k\theta_j, \omega_i), \nabla_l \tilde{f}(x_k + c_k\theta_j, \omega_i))] \\
=& [\frac{\alpha_0}{n2^d(\alpha_0 c_k^2 t_l^2 + \alpha_l)}]^2 \sum_{j=1}^{2^d}\sum_{i=1}^{n} [c_k^2 t_l^2 \sigma_f^2 + (\frac{\alpha_l}{\alpha_0})^2 \sigma_{g,l}^2 + 2c_k\theta_{j,l}\frac{\alpha_l}{\alpha_0}\sigma_{f/g,l}^2] \\
=& [\frac{\alpha_0}{n2^d(\alpha_0 c_k^2 t_l^2 + \alpha_l)}]^2 \sum_{j=1}^{2^d}\sum_{i=1}^{n} [c_k^2 t_l^2 \sigma_f^2 + (\frac{\alpha_l}{\alpha_0})^2 \sigma_{g,l}^2],
\end{aligned}$$

where the last equation follows from the symmetric property of $\theta_{j,l}$ and $\theta_{j,l} \in \{-t_l, t_l\}$.

Therefore,

$$\begin{aligned}
\mathrm{Var}(\hat{\beta}_{k,l}) =& \frac{1}{[n2^d(\alpha_0 c_k^2 t_l^2 + \alpha_l)]^2} \sum_{j=1}^{2^d}\sum_{i=1}^{n} [\alpha_0^2 c_k^2 t_l^2 \sigma_f^2 + \alpha_l^2 \sigma_{g,l}^2] \\
=& \frac{1}{n2^d(\alpha_0 c_k^2 t_l^2 + \alpha_l)^2} (\alpha_0^2 c_k^2 t_l^2 \sigma_f^2 + \alpha_l^2 \sigma_{g,l}^2) \\
=& \frac{1}{n2^d(c_k^2 t_l^2 + r_l)^2} (c_k^2 t_l^2 \sigma_f^2 + r_l^2 \sigma_{g,l}^2),
\end{aligned}$$

where $r_l = \alpha_l/\alpha_0$. Taking derivative w.r.t. r_l,

$$\frac{d\mathrm{Var}(\hat{\beta}_{k,l})}{dr_l} = \frac{1}{n2^d(c_k^2 t_l^2 + r_l)^3} (2c_k^2 t_l^2 \sigma_{g,l}^2 r_l - 2c_k^2 t_l^2 \sigma_f^2). \tag{2.12}$$

Setting the derivative in Equation (2.12) to 0 yields the optimal weight ratio Equation (2.9).

Combining the normalizing condition (i.e., $\sum_{i=0}^{d} \alpha_i = 1$) yields Equations (2.10) and (2.11).

Since $\frac{d^2\text{Var}(\hat{\beta}_{k,l})}{dr_l^2} > 0$ when evaluated at r_l^*, the proof is complete.

□

Remark 2.2. *We do not make any assumption on how the randomness comes into the measurement (e.g., additive, multiplicative, etc.) for the weight tuning. We only assume that the measurement noises of responses and gradients are sample-wise independent.*

Remark 2.3. *When the actual variances of measurement noises are unknown, they can be approximated by sample variances, which can be applied to estimate the optimal weights. We evaluate weights estimated using sample variances in Section 2.6.*

2.4 DiGARSM with Simultaneous Perturbation (SP-DiGARSM)

One limitation of DiGARSM with full factorial design is that the sampling and computation effort required is exponential in the number of dimensions, thus making DiGARSM impractical for solving problems in high dimensions. Therefore, we consider employing the simultaneous perturbation technique in DiGARSM (SP-DiGARSM), which only requires two gradient and response measurements per iteration. Specifically, let $\Delta_k \in \mathbb{R}^d$ be a vector of d i.i.d. zero-mean random variables. At iteration k, SP-DiGARSM obtains response samples $\tilde{f}(x_k + c_k\Delta_k, \omega_i)$, $\tilde{f}(x_k - c_k\Delta_k, \omega_i)$ and direct gradient samples $\widetilde{\nabla f}(x_k + c_k\Delta_k, \omega_i)$, $\widetilde{\nabla f}(x_k - c_k\Delta_k, \omega_i)$ for $i = 1, 2, \ldots, n$.

Similar to DiGARSM, fit the samples to the augmented linear model in Equations (2.5) and (2.6) by minimizing the loss function

$$L = \alpha_0 \sum_{i=1}^{n} [(\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \hat{f}(x_k + c_k\Delta_k))^2 + (\tilde{f}(x_k - c_k\Delta_k, \omega_i) - \hat{f}(x_k - c_k\Delta_k))^2] +$$
$$\sum_{i=1}^{n} (\widetilde{\nabla f}(x_k + c_k\Delta_k, \omega_i) - \widehat{\nabla f}(x_k + c_k\Delta_k))^T W (\widetilde{\nabla f}(x_k + c_k\Delta_k, \omega_i) - \widehat{\nabla f}(x_k + c_k\Delta_k)) +$$
$$\sum_{i=1}^{n} (\widetilde{\nabla f}(x_k - c_k\Delta_k, \omega_i) - \widehat{\nabla f}(x_k - c_k\Delta_k))^T W (\widetilde{\nabla f}(x_k - c_k\Delta_k, \omega_i) - \widehat{\nabla f}(x_k - c_k\Delta_k)).$$

The minimizing β_k (therefore g_k) for SP-DiGARSM is given by

$$\hat{\beta}_k = [\alpha_0 \sum_{i=1}^{n} ((x_k + c_k\Delta_k - \bar{x}_k)(x_k + c_k\Delta_k - \bar{x}_k)^T + (x_k - c_k\Delta_k - \bar{x}_k)(x_k - c_k\Delta_k - \bar{x}_k)^T) + 2nW]^{-1}$$
$$[\alpha_0 \sum_{i=1}^{n} ((x_k + c_k\Delta_k - \bar{x}_k)(\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \bar{\tilde{f}}_k) + (x_k - c_k\Delta_k - \bar{x}_k)(\tilde{f}(x_k - c_k\Delta_k, \omega_i) - \bar{\tilde{f}}_k)) +$$
$$2nW\overline{\widetilde{\nabla f_k}}]. \tag{2.13}$$

Similar to DiGARSM, when $\alpha_0 = 0$, only gradient information is utilized. On the other hand, setting $\alpha_0 = 1$ (i.e., W is a matrix of 0's and gradient information is unused) leads to an infinite number of solutions to $\hat{\beta}_k$.

An algorithmic description of SP-DiGARSM is presented in Algorithm 2.

2.5 Convergence Analysis

In this section, we present convergence theorems for both DiGARSM and SP-DiGARSM. Two types of convergence are established, i.e., almost sure convergence and mean-squared convergence, where a convergence rate analysis is provided for the latter

Algorithm 2: SP-DiGARSM Stochastic Approximation.

Input: Initial point x_0, weights $\alpha_0, \ldots, \alpha_d$, positive sequences $\{a_k\}$ and $\{c_k\}$, uniformly bounded zero mean input distribution of Δ_k, number of samples/replications for each point n

Output: Optimal point $x^* = \arg\min_x f(x)$

$k \leftarrow 0$
while *stopping rule not met* **do**
  Generate i.i.d. samples $\{\Delta_{k,i}\}_{i=1}^{d}$, $i = 1, 2, \ldots, d$
  for $i = 1, 2, \ldots, n$ **do**
    Obtain samples of response and gradient:
    $\tilde{f}(x_k + c_k\Delta_k, \omega_i)$ and $\widetilde{\nabla f}(x_k + c_k\Delta_k, \omega_i)$
    $\tilde{f}(x_k - c_k\Delta_k, \omega_i)$ and $\widetilde{\nabla f}(x_k - c_k\Delta_k, \omega_i)$
  end
  Calculate gradient estimate g_k by Equation (2.13)
  $x_{k+1} \leftarrow x_k - a_k g_k$
  $k \leftarrow k + 1$
end
return x_k

under different choices of a_k and c_k. The proof for DiGARSM is similar to that of SP-DiGARSM and can be carried out analogously, which is therefore omitted for clarity.

Define the bias and error of gradient estimate $\hat{\beta}_k$, respectively, by

$$b(x_k) = \mathbb{E}[\hat{\beta}_k - \nabla f(x_k) | x_k], \tag{2.14}$$

$$e_k = \hat{\beta}_k - \mathbb{E}[\hat{\beta}_k | x_k]. \tag{2.15}$$

For the sake of presentation, for SP-DiGARSM, define

$$D_k = \Delta_k \Delta_k^T,$$

$$M_k = (\alpha_0 c_k^2 D_k + W)^{-1}.$$

We first prove that the estimators provided by DiGARSM and SP-DiGARSM are asymptotically unbiased. Then we apply results from [39] and [40] to establish conver-

gence with appropriate regularity assumptions.

2.5.1 Convergence of Stochastic Approximation with SP-DiGARSM

Lemma 2.1 (SP-DiGARSM asymptotic unbiasedness)**.** *Suppose* $\{x_k\}$ *is generated via recursion Equation (2.2) using the SP-DiGARSM gradient estimator given by Equation (2.13). If*

1. $\{\Delta_{k,j}\}_{j=1}^{d}$ *are symmetrically i.i.d. with mean zero, uniformly bounded with finite inverse moments, i.e., there exists* $K_0 > 0$ *such that* $0 < |\Delta_{k,j}| \leq K_0 \ \forall k, j$*, and are independent from the response and gradient measurement noise,*

2. f *is three-times differentiable, and* $|f_{i,j}^{(3)}(\cdot)|$ *is uniformly bounded by* $K_1 > 0$*,*

3. *the positive sequence* $\{c_k\}$ *converges to 0 in the limit, i.e.,* $\lim_{k\to\infty} c_k = 0$*, and*

4. *the measurement noise is additive for both gradient and response with mean 0, i.e.,* $\tilde{f}(x) = f(x) + \varepsilon$*,* $\widetilde{\nabla f}(x) = \nabla f(x) + \delta$*,*

then the stochastic gradient estimator provided by Equation (2.13) is asymptotically unbiased, i.e.,

$$b(x_k) \to 0 \text{ at the rate of } \mathcal{O}(c_k^2) \text{ as } k \to \infty.$$

Proof.

$$\hat{\beta}_k = [2\alpha_0 \sum_{i=1}^{n} (c_k\Delta_k)(c_k\Delta_k)^T + 2nW]^{-1}$$
$$[\alpha_0 \sum_{i=1}^{n} (c_k\Delta_k)(\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \alpha_0 \sum_{i=1}^{n} (c_k\Delta_k)(\tilde{f}(x_k - c_k\Delta_k, \omega_i) - \bar{\tilde{f}}_k) + 2nW\widetilde{\overline{\nabla f_k}}].$$

Note that

$$2\alpha_0 \sum_{i=1}^{n} (c_k\Delta_k)(c_k\Delta_k)^T = 2n\alpha_0 c_k^2 D_k,$$

where $D_k = \Delta_k\Delta_k^T$, and $\bar{\tilde{f}}_k$ is a constant with respect to i. Then, the estimator can be rewritten as

$$\hat{\beta}_k = (2n\alpha_0 c_k^2 D_k + 2nW)^{-1}[\alpha_0 c_k\Delta_k \sum_{i=1}^{n} (\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \tilde{f}(x_k - c_k\Delta_k, \omega_i)) + 2nW\widetilde{\overline{\nabla f_k}}]$$
$$= \frac{1}{2n} M_k[\alpha_0 c_k\Delta_k \sum_{i=1}^{n} (\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \tilde{f}(x_k - c_k\Delta_k, \omega_i)) + 2nW\widetilde{\overline{\nabla f_k}}],$$

where $M_k = (\alpha_0 c_k^2 D_k + W)^{-1}$.

For ease of presentation, let

$$\hat{\beta}_k^1 = \frac{1}{2n} M_k \alpha_0 c_k\Delta_k \sum_{i=1}^{n} (\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \tilde{f}(x_k - c_k\Delta_k, \omega_i)),$$
$$\hat{\beta}_k^2 = \frac{1}{2n} M_k 2nW\widetilde{\overline{\nabla f_k}} = M_k W\widetilde{\overline{\nabla f_k}}.$$

We first consider $\hat{\beta}_k^1$.

$$\begin{aligned}\mathbb{E}[\hat{\beta}_k^1|x_k] =&\mathbb{E}[\frac{1}{2n}M_k\alpha_0 c_k\Delta_k\sum_{i=1}^{n}(\tilde{f}(x_k+c_k\Delta_k,\omega_i)-\tilde{f}(x_k-c_k\Delta_k,\omega_i))|x_k]\\ =&\mathbb{E}[\frac{1}{2n}M_k\alpha_0 c_k\Delta_k\sum_{i=1}^{n}(f(x_k+c_k\Delta_k)-f(x_k-c_k\Delta_k))|x_k],\end{aligned}$$

where the second equation follows from the additive noise assumption.

Using a Taylor's series expansion, we have

$$f(x_k\pm c_k\Delta_k) = f(x_k)\pm c_k\langle\nabla f(x_k),\Delta_k\rangle+\frac{c_k^2}{2}\langle\nabla^2 f(x_k)\Delta_k,\Delta_k\rangle\pm\frac{c_k^3}{6}\nabla^3 f(t_k^{\pm})(\Delta_k\otimes\Delta_k\otimes\Delta_k),$$

where $t_k^{\pm}$ are on the line segment between x_k and $x_k\pm c_k\Delta_k$. Therefore,

$$\begin{aligned}\mathbb{E}[\hat{\beta}_k^1|x_k] =&\mathbb{E}[\frac{1}{2n}M_k\alpha_0 c_k\Delta_k\sum_{i=1}^{n}(f(x_k+c_k\Delta_k)-f(x_k-c_k\Delta_k))|x_k]\\ =&\mathbb{E}[M_k\alpha_0 c_k^2\Delta_k\langle\nabla f(x_k),\Delta_k\rangle+\frac{1}{12}M_k\alpha_0 c_k^4\Delta_k(\nabla^3 f(t_k^+)-\nabla^3 f(t_k^-))(\Delta_k\otimes\Delta_k\otimes\Delta_k)|x_k]\\ =&\mathbb{E}[M_k\alpha_0 c_k^2\Delta_k\Delta_k^T\nabla f(x_k)+\frac{1}{12}M_k\alpha_0 c_k^4\Delta_k(\nabla^3 f(t_k^+)-\nabla^3 f(t_k^-))(\Delta_k\otimes\Delta_k\otimes\Delta_k)|x_k].\end{aligned}$$

Now we consider $\mathbb{E}[\hat{\beta}_k^2|x_k]$.

$$\mathbb{E}[\hat{\beta}_k^2|x_k] = M_k W\widetilde{\nabla f_k}M_k W\frac{1}{2}(\nabla f(x_k+c_k\Delta_k)+\nabla f(x_k-c_k\Delta_k)).$$

Similarly, the $\nabla f(x_k+c_k\Delta_k)$ can be expressed by

$$\nabla f(x_k\pm c_k\Delta_k) = \nabla f(x_k)\pm c_k\nabla^2 f(x_k)\Delta_k+\frac{c_k^2}{2}\nabla^3 f(s_k^{\pm})(\Delta_k\otimes\Delta_k),$$

where $s_k^{\pm}$ are on the line segment between x_k and $x_k \pm c_k\Delta_k$. Therefore,

$$\mathbb{E}[\hat{\beta}_k^2|x_k] = \mathbb{E}[M_k W \nabla f(x_k) + \frac{c_k^2}{2} M_k W(\nabla^3 f(s_k^+) + \nabla^3 f(s_k^-))(\Delta_k \otimes \Delta_k)|x_k].$$

Combining the two terms yields

$$\begin{aligned}
b(x_k) =& \mathbb{E}[\hat{\beta}_k - \nabla f(x_k)|x_k] \\
=& \mathbb{E}[M_k \alpha_0 c_k^2 \Delta_k \Delta_k^T \nabla f(x_k) + M_k W \nabla f(x_k) - \nabla f(x_k) \\
&+ \frac{1}{12} M_k \alpha_0 c_k^4 \Delta_k (\nabla^3 f(t_k^+) - \nabla^3 f(t_k^-))(\Delta_k \otimes \Delta_k \otimes \Delta_k) \\
&+ \frac{c_k^2}{2} M_k W(\nabla^3 f(s_k^+) + \nabla^3 f(s_k^-))(\Delta_k \otimes \Delta_k)|x_k] \\
=& c_k^2 \mathbb{E}[\frac{1}{12} M_k \alpha_0 c_k^2 \Delta_k (\nabla^3 f(t_k^+) - \nabla^3 f(t_k^-))(\Delta_k \otimes \Delta_k \otimes \Delta_k) \\
&+ \frac{1}{2} M_k W(\nabla^3 f(s_k^+) + \nabla^3 f(s_k^-))(\Delta_k \otimes \Delta_k)|x_k] \\
=& c_k^2 \mathbb{E}[M_k \alpha_0 c_k^2 D_k \frac{\Delta_k}{12\Delta_k^T \Delta_k} (\nabla^3 f(t_k^+) - \nabla^3 f(t_k^-))(\Delta_k \otimes \Delta_k \otimes \Delta_k) \\
&+ M_k W \frac{\nabla^3 f(s_k^+) + \nabla^3 f(s_k^-)}{2} (\Delta_k \otimes \Delta_k)|x_k].
\end{aligned}$$

Denote the bias of the l-th element by $b_l(x_k)$. Under Assumptions 1 and 2 and using similar analysis as that in Lemma 1 of [38], there exist positive number K that upper bounds each element of $\frac{\Delta_k}{12\Delta_k^T\Delta_k}(\nabla^3 f(t_k^+) - \nabla^3 f(t_k^-))(\Delta_k \otimes \Delta_k \otimes \Delta_k)$ and $\frac{\nabla^3 f(s_k^+) + \nabla^3 f(s_k^-)}{2}(\Delta_k \otimes \Delta_k)$.

Therefore,

$$
\begin{aligned}
b_l(x_k) =& c_k^2 \mathbb{E}[\{\alpha_0 c_k^2 M_k D_k \frac{\Delta_k}{12\Delta_k^T\Delta_k}(\nabla^3 f(t_k^+) - \nabla^3 f(t_k^-))(\Delta_k \otimes \Delta_k \otimes \Delta_k) \qquad (2.16)\\
&+ M_k W \frac{\nabla^3 f(s_k^+) + \nabla^3 f(s_k^-)}{2}(\Delta_k \otimes \Delta_k)\}_l | x_k] \\
\leq& c_k^2 \mathbb{E}[\{\alpha_0 c_k^2 M_k D_k K \mathbf{1}^T + M_k W K \mathbf{1}^T\}_l | x_k] \\
\leq& c_k^2 K, \qquad (2.17)
\end{aligned}
$$

where $\mathbf{1}^T = [1, 1, \ldots, 1]$. By Assumption 3, $b_l(x_k) \to 0$ as $k \to \infty$ as desired. □

With Lemma 2.1, we are ready to state our main convergence theorems. We show that, under different conditions, the proposed (SP-)DiGARSM converges to the optimum with probability 1 (Theorem 2.1) and in mean square (Theorem 2.2).

Theorem 2.1. *Let $\{x_k\}$ be a sequence generated via recursion Equation (2.2) using SP-DiGARSM with gradient estimator given by Equation (2.13). If the conditions from Lemma 2.1 in addition to the following are satisfied:*

(A1) there exist positive constants K_2 and K_3 such that $\mathbb{E}[(\tilde{f}(x_k \pm c_k\Delta_k))^4] \leq K_2$ and $\mathbb{E}[(\nabla_l f(x_k + c_k\Delta_k)^4)] \leq K_3$ for all $k > 0$ and $l = 1, 2, \ldots, d$,

(A2) There exist positive sequences $\{a_k\}$ and $\{c_k\}$ such that $c_k \to 0$ as $n \to \infty$, $\sum_{k=1}^{\infty} a_k = \infty$ and $\sum_{k=1}^{\infty} a_k^2 < \infty$;

(A3) for all n, $\|x_n\| < \infty$ a.s.;

(A4) x^* is an asymptotically stable solution of the differential equation $\frac{dx(t)}{dt} = -\nabla f(x)$; and

(A5) for $D(x^*) = \{x_0 : \lim_{t\to\infty} x(t|x_0) = x^*\}$ *where* $x(t|x_0)$ *denotes the solution to the differential equation of (A4) with initial condition* x_0 *(i.e.,* $D(x^*)$ *is the domain of attraction), there exists a compact* $S \subseteq D(x^*)$ *such that* $x_k \in S$ *infinitely often for almost all sample points;*

then, as $k \to \infty$,

$$x_k \to x^* \ a.s. \tag{2.18}$$

Proof. First note that the largest eigenvalue of matrix M_k (denoted by λ_m) bounded by $\min(K_4 c_k^{-2}, 1/\lambda_m^W)$, where $K_4 > 0$ and λ_m^W is the largest eigenvalue of W. Since matrices D_k and W are symmetric and positive semi-definite, by Courant–Fischer–Weyl min-max principle (`https://en.wikipedia.org/wiki/Weyl%27s_inequality`), we have

$$eig(\alpha_0 c_k^2 D_k + W) \geq \max(eig(\alpha_0 c_k^2 D_k), eig(W)) \geq \max(\alpha_0 \|\Delta_k\|^2 c_k^2, \lambda_m^W).$$

Therefore,

$$\lambda_m \leq \min(1/(\alpha_0 \|\Delta_k\|^2 c_k^2), 1/\lambda_m^W) = \min(K_4 c_k^{-2}, 1/\lambda_m^W).$$

Since $c_k \to 0$ as $k \to \infty$, λ_m will be bounded by $1/\lambda_m^W$ for k sufficiently large. From Lemma 2.2.1 and Theorem 2.3.1 in [39], if (A2) to (A5) are satisfied, Equation (2.18) holds if

(a) $\|b(x_k)\| < \infty$ for all k and $b(x_k) \to 0$ as $k \to \infty$ a.s., and

(b) $\lim_{k\to\infty} P(\sup_{m\geq k} \|\sum_{q=k}^{m} a_q e_q\| \geq \eta) = 0$ for all $\eta > 0$.

(a) follows directly from Lemma 2.1.

For (b), we start from proving that

$$\sum_{q=k}^{\infty} a_q^2 \mathbb{E}[\|e_q\|^2] < \infty. \tag{2.19}$$

The l-th component of the variance of gradient estimate is bounded by

$$\mathbb{E}[(\hat{\beta}_{k,l})^2] = \mathbb{E}[(\beta_{k,l}^1 + \beta_{k,l}^2)^2],$$

where

$$\hat{\beta}_{k,l}^1 = \frac{\alpha_0 c_k}{2n} e_l^T M_k \Delta_k \sum_{i=1}^{n} (\tilde{f}(x_k + c_k\Delta_k, \omega_i) - \tilde{f}(x_k - c_k\Delta_k, \omega_i)),$$

$$\hat{\beta}_{k,l}^2 = e_l^T M_k W \widetilde{\nabla f_k},$$

where e_l denotes the unit vector in the l-th direction. Apply inequality

$$\Big[\frac{1}{n}\sum_{i=1}^{n} a_i\Big]^2 \leq \frac{1}{n}\sum_{i=1}^{n} a_i^2$$

then

$$\mathbb{E}[(\hat{\beta}_{k,l})^2] \leq 2\mathbb{E}[(\beta_{k,l}^1)^2 + (\beta_{k,l}^2)^2].$$

Apply Cauchy-Schwartz inequality,

$$\begin{aligned}
\mathbb{E}[(\beta^1_{k,l})^2] =& \alpha_0^2 c_k^2 \mathbb{E}[(e_l^T M_k \Delta_k)^2 (\frac{1}{2n}\sum_{i=1}^{n} \tilde{f}(x_k + c_k\Delta_k, \omega_i) - \tilde{f}(x_k - c_k\Delta_k, \omega_i))^2] \\
\leq& \alpha_0^2 c_k^2 \mathbb{E}[(e_l^T M_k \Delta_k)^2 (\frac{1}{2n}\sum_{i=1}^{n} \tilde{f}(x_k + c_k\Delta_k, \omega_i)^2 + \tilde{f}(x_k - c_k\Delta_k, \omega_i)^2)] \\
\leq& \alpha_0^2 c_k^2 \mathbb{E}[(e_l^T M_K \Delta_k)^4]^{\frac{1}{2}} \mathbb{E}[(\frac{1}{2n}\sum_{i=1}^{n} \tilde{f}(x_k + c_k\Delta_k, \omega_i)^2 + \tilde{f}(x_k - c_k\Delta_k, \omega_i)^2)^2]^{\frac{1}{2}} \\
\leq& \alpha_0^2 c_k^2 \mathbb{E}[(e_l^T M_k \Delta_k)^4]^{\frac{1}{2}} \mathbb{E}[(\frac{1}{2n}\sum_{i=1}^{n} \tilde{f}(x_k + c_k\Delta_k, \omega_i)^4 + \tilde{f}(x_k - c_k\Delta_k, \omega_i)^4)]^{\frac{1}{2}} \\
\leq& K_2^2 \alpha_0^2 c_k^2 \mathbb{E}[(e_l^T M_K \Delta_k)^4]^{\frac{1}{2}}.
\end{aligned}$$

Since

$$\mathbb{E}[(e_l^T M_K \Delta_k)^4] \leq \mathbb{E}[(e_l^T M_K e_l)^2 (\Delta_k^T M_K \Delta_k)^2] \leq \lambda_m^4 \mathbb{E}[\|\Delta_k\|_2^4] \leq d^2 \lambda_m^4 K_0^4$$

$\mathbb{E}[(\beta^1_{k,l})^2]$ is bounded by

$$\mathbb{E}[(\beta^1_{k,l})^2] \leq d\lambda_m^2 K_0^2 K_2^2 \alpha_0^2 c_k^2. \tag{2.20}$$

Similarly,

$$\begin{aligned}
\mathbb{E}[(\beta_{k,l}^2)^2] =& \mathbb{E}[(e_l^T M_K W \overline{\widetilde{\nabla f_k}})^2] \\
\leq& \mathbb{E}[(e_l^T M_K \overline{\widetilde{\nabla f_k}})^2] \\
\leq& \mathbb{E}[(e_l^T M_K e_l)(\overline{\widetilde{\nabla f_k}}^T M_K \overline{\widetilde{\nabla f_k}})] \\
=& \lambda_m^2 \mathbb{E}[\|\overline{\widetilde{\nabla f_k}}\|_2^2] \\
=& \lambda_m^2 \mathbb{E}[\sum_{l=1}^{d} (\frac{1}{2n} \sum_{i=1}^{n} (\widetilde{\nabla f}(x_k + c_k\Delta_k, \omega_i) + \widetilde{\nabla f}(x_k - c_k\Delta_k, \omega_i)))^2] \\
\leq& \frac{1}{2n} \lambda_m^2 \mathbb{E}[\sum_{l=1}^{d} \sum_{i=1}^{n} (\widetilde{\nabla f}(x_k + c_k\Delta_k, \omega_i)^2 + \widetilde{\nabla f}(x_k - c_k\Delta_k, \omega_i)^2)] \\
\leq& d\lambda_m^2 K_3^2.
\end{aligned}$$

Thus, the variance is bounded by

$$\mathbb{E}[(\hat{\beta}_{k,l})^2] \leq 2(d\lambda_m^2 K_0^2 K_2^2 \alpha_0^2 c_k^2 + d\lambda_m^2 K_3^2 K_4^2) \leq dK_5/(\lambda_m^W)^2 \tag{2.21}$$

for some constant $K_5 > 0$, as $c_k \to 0$. Since

$$\begin{aligned}
\mathbb{E}[\|e_k\|^2] = \mathbb{E}[e_k^T e_k] &= \mathbb{E}[(\hat{\beta}_k - \mathbb{E}[\hat{\beta}_k|x_k])^T (\hat{\beta}_k - \mathbb{E}[\hat{\beta}_k|x_k])] \\
&= \mathbb{E}[\hat{\beta}_k^T \hat{\beta}_k] - \mathbb{E}[\mathbb{E}[\hat{\beta}_k|x_k]^T \mathbb{E}[\hat{\beta}_k|x_k]] \leq \mathbb{E}[\hat{\beta}_k^T \hat{\beta}_k],
\end{aligned}$$

the squared norm of error at iteration k is bounded by

$$\mathbb{E}[\|e_k\|^2] \leq \mathbb{E}[\|\hat{\beta}_k\|^2] \leq 2d^2(\lambda_m^2 K_0^2 K_2^2 \alpha_0^2 c_k^2 + K_3^2 K_4^2 c_k^{-4}). \tag{2.22}$$

By Inequality (2.22) and (A2), Inequality (2.19) holds. Apply Doob's inequality to the martingale sequence $\{\sum_{q=n}^{m} a_q e_q(x_q)\}_{m\geq n}$ and Fubini's theorem to obtain,

$$P(\sup_{m\geq k} \| \sum_{q=k}^{m} a_q e_q \| \geq \eta) \leq \eta^{-2}\mathbb{E}[\| \sum_{q=k}^{\infty} a_q e_q \|^2] = \eta^{-2} \sum_{q=k}^{\infty} a_q^2 \mathbb{E}[\|e_q\|^2],$$

where the equality follows from

$$\mathbb{E}[e_p^T e_q] = \mathbb{E}[\mathbb{E}[e_p^T e_q | x_q]] = \mathbb{E}[e_p^T \mathbb{E}[e_q | x_q]] = 0, \ \forall p < q.$$

Since $\sum_{q=k}^{\infty} a_q^2 \mathbb{E}[\|e_q\|^2]$ converges, (b) holds, which completes the proof. □

Under different conditions, it can be shown that SP-DiGARSM converges in mean square at the rate of $O(k^{-1})$:

Theorem 2.2. *Let $\{x_k\}$ be a sequence generated via recursion Equation (2.2) using SP-DiGARSM with gradient estimator given by Equation (2.13). If conditions from Lemma 2.1 in addition to the following are satisfied:*

1. *The objective function $f(x)$ is strongly convex, i.e., there exists a positive number μ such that*

$$f(y) \geq f(x) + (y-x)^T \nabla f(x) + \frac{1}{2}\mu \|y-x\|^2;$$

and

2. *the stepsize a_k and finite difference c_k have form $a_k = a_0/(1+k+A)^\alpha$ and $c_k = c_0/(1+k)^\gamma$, respectively, where $a_0, c_0, \alpha, \gamma > 0$, $A \geq 0$, $2a\mu > 1-2\gamma$ and A, a_0, α*

are chosen such that $1 - 2\mu a_1 > 0$.

Then, x_k *converges to* x^* *in mean square at the rate of*

$$\mathbb{E}[\|x_k - x^*\|^2] = \begin{cases} \mathcal{O}(k^{-\alpha}) & \text{if } \gamma \geq \alpha/4, \\ \mathcal{O}(k^{-4\gamma}) & \text{if } \gamma < \alpha/4. \end{cases}$$

We provide a sketch of proof in this dissertation. The derivations can be carried out using similar arguments as that in [40], therefore we refer readers to their paper for details.

Proof. Denote the expected squared error at iteration k by

$$E_k = \mathbb{E}[\|x_k - x^*\|^2],$$

then, we can write

$$\begin{aligned} E_{k+1} =& \mathbb{E}[\|x_k - a_k\hat{\beta}_k - x^*\|^2] \\ =& E_k + a_k^2\mathbb{E}[\|\hat{\beta}_k\|^2] - 2a_k\mathbb{E}[(x_k - x^*)^T\hat{\beta}_k], \end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[(x_k - x^*)^T \hat{\beta}_k] =& \mathbb{E}[(x_k - x^*)^T (\hat{\beta}_k - \nabla f(x_k))] + \mathbb{E}[(x_k - x^*)^T \nabla f(x_k)] \\ \geq& \mathbb{E}[\mathbb{E}[(x_k - x^*)^T (\hat{\beta}_k - \nabla f(x_k))|x_k]] + \mu E_k \text{ (strong convexity)} \\ =& \mathbb{E}[(x_k - x^*)^T b(x_k)] + \mu E_k \\ \geq& -\mathbb{E}[\|(x_k - x^*)\| \cdot \|b(x_k)\|] + \mu E_k \text{ (Cauchy-Schwartz inequality) .}\end{aligned}$$

Therefore,

$$E_{k+1} \leq (1 - 2a_k\mu)E_k + a_k^2 \mathbb{E}[\|\hat{\beta}_k\|^2] + 2a_k \mathbb{E}[(x_k - x^*)^T b(x_k)].$$

Since $1 - 2\mu a_k < 1 \ \forall k \geq 1$, it can be shown that

$$E_{k+1} \leq T_k E_1 + T_k \sum_{i=1}^{k} \frac{a_i^2}{T_i} \mathbb{E}[\|\hat{\beta}_i\|^2] + T_k \sum_{i=1}^{k} \frac{2a_i}{T_i} \mathbb{E}[(x_i - x^*)^T b(x_i)], \tag{2.23}$$

where $T_k = \prod_{i=1}^{k}(1 - 2\mu a_i)$.

Since $\mathbb{E}[\|\hat{\beta}_k\|^2] = \mathcal{O}(1)$ (from inequality (2.21)) and $\|b(x_k)\| = \mathcal{O}(c_k^2)$ (from inequality (2.17)), by similar arguments of Theorem B.1 in [40], E_k converges in mean square, and if we assume E_k converges at the rate of $\mathcal{O}(k^{-2t})$ with unknown $t > 0$, the convergence rate of each term in Inequality (2.23) is summarized in Table 2.1, where the convergence rate of E_k will be the slowest rate of the three terms.

The next step is to solve t. Since we assume $1 - 2a_k\mu > 0, \ \forall k$, the first term will converge faster than the first and second term. For $\frac{1}{2} < \alpha < 1$, suppose the third term

	$\alpha = 1$	$\frac{1}{2} < \alpha < 1$
first term	$\mathcal{O}(k^{-2a_0\mu})$	$\mathcal{O}(\exp(-2a_0\mu(1+k+A)^{1-\alpha}/(1-\alpha)))$
second term	$\mathcal{O}(k^{-1})$	$\mathcal{O}(k^{-\alpha})$
third term	$\mathcal{O}(k^{-(t+2\gamma)})$	$\mathcal{O}(k^{-(t+2\gamma)})$
E_k	$\mathcal{O}(k^{-2t})$	$\mathcal{O}(k^{-2t})$

Table 2.1: Convergence rate of each term in Inequality (2.23)

converges faster than the second term, i.e., $t+2\gamma \geq \alpha$, then $2t = \alpha$ and $\gamma \geq \alpha/4$. In this case, $t = \alpha/2$ and E_k converges at the rate of $\mathcal{O}(k^{-\alpha})$. Similarly, suppose the second term converges faster than the third term, i.e., $t+2\gamma < \alpha$, then $2t = t+2\gamma$ and $\gamma < \alpha/4$. In this case, $t = 2\gamma$ and E_k converges at the rate of $\mathcal{O}(k^{-4\gamma})$. Similar analysis can be done for the $\alpha = 1$ case. In summary, we have

$$\mathbb{E}[\|x_k - x^*\|^2] = \begin{cases} \mathcal{O}(k^{-\alpha}) & \text{if } \gamma \geq \alpha/4, \\ \mathcal{O}(k^{-4\gamma}) & \text{if } \gamma < \alpha/4, \end{cases}$$

with optimal rate achieved at $\mathcal{O}(k^{-1})$ at $\alpha = 1$ and $\gamma \geq 1/4$.

□

From Theorem 2.2, the optimal convergence rate of SP-DiGARSM can be achieved at $\mathcal{O}(k^{-1})$ when $\alpha = 1$ and $\gamma \geq 1/4$.

Remark 2.4. *When only response information is used, algorithms such as SPSA can only achieve $\mathcal{O}(k^{-2/3})$ mean square convergence. This theorem shows in theory how the additional gradient would improve the convergence rate.*

2.5.2 Convergence of Stochastic Approximation with DiGARSM

The proof of the convergence of DiGARSM can be carried out analogously with weaker assumptions, as there is less randomness in the estimator.

Lemma 2.2 (DiGARSM asymptotic unbiasedness)**.** *Suppose* $\{x_k\}$ *is generated via recursion Equation (2.2) using DiGARSM gradient estimator given by Equation (2.8). Under Assumptions 1 to 3 in Lemma 2.1, the stochastic gradient estimator provided by Equation (2.8) is asymptotically unbiased, i.e.,*

$$b(x_k) \to 0 \text{ at the rate of } \mathcal{O}(c_k^2) \text{ as } k \to \infty.$$

Remark 2.5. *For DiGARSM, we can drop the additive noise assumption. Because unlike* Δ_k*, which is a random variable,* θ_j *is a fixed vector for given* j*. Therefore the matrix* $[\alpha_0 \sum_{j=0}^{2^d} \sum_{i=1}^{n} (c_k\theta_j)(c_k\theta_j)^T + 2^d nW]$ *in Equation (2.8) is a constant w.r.t.* $\tilde{f}(x)$ *and* $\widetilde{\nabla f}(x)$*, and thus can be decoupled when taking expectation.*

Theorem 2.3. *Let* $\{x_k\}$ *be a sequence generated via recursion Equation (2.2) using DiGARSM with gradient estimator given by Equation (2.8). If the conditions from Lemma 2.2 and Assumptions from Theorem 2.1 are satisfied, then, as* $k \to \infty$

$$x_k \to x^* \ a.s. \tag{2.24}$$

Theorem 2.4. *Let* $\{x_k\}$ *be a sequence generated via recursion Equation (2.2) using DiGARSM with gradient estimator given by Equation (2.8). If conditions from Lemma 2.2*

and Theorem 2.2 are satisfied, then, x_k converges to x^* in mean square at the rate of

$$\mathbb{E}[\|x_k - x^*\|^2] = \begin{cases} \mathcal{O}(k^{-\alpha}) & \text{if } \gamma \geq \alpha/4, \\ \mathcal{O}(k^{-4\gamma}) & \text{if } \gamma < \alpha/4, \end{cases}$$

with optimal rate achieved at $\mathcal{O}(k^{-1})$ at $\alpha = 1$ and $\gamma \geq 1/4$.

The proofs of Lemma 2.2 and Theorem 2.3 are similar to that of SP-DiGARSM, and thus omitted here.

2.6 Numerical Experiments

Three experiments are used to evaluate the efficiency of DiGARSM, SP-DiGARSM and optimal weights for DiGARSM. All experiments are run on a test function (also known as the Trid function) defined by

$$f(x) = \sum_{i=1}^{d} (x^i - 1)^2 - \sum_{i=2}^{d} x^i x^{i-1}, \tag{2.25}$$

where x^i is the i-th element of x, with the gradient of the i-th dimension given by

$$\nabla_i f(x) = 2(x^i - 1) - x^{i-1}\mathbb{1}\{i > 1\} - x^{i+1}\mathbb{1}\{i < d\}, \tag{2.26}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function. The minimum is reached at $x^{i,*} = i(d+1-i)$ for $i = 1,2,\ldots,d$. We assume additive homogeneous Gaussian noise with zero mean, i.e.,

$$\begin{aligned}\tilde{f}(x) =& f(x) + \varepsilon,\\ \widetilde{\nabla f}(x) =& \nabla f(x) + \delta,\end{aligned}$$

where

$$\begin{aligned}&\varepsilon \sim N(0, \sigma_f^2),\\ &\delta \sim N(\mathbf{0}, \Sigma),\ \Sigma_{l,m} = \begin{cases} \sigma_{g,l}^2,\ l = m.\\ 0,\ otherwise.\end{cases}\end{aligned}$$

Unless otherwise noted, the following applies for all experiments:

1. $\sigma_f^2 = 40$, $\sigma_{g,l}^2 = 40$ and $\sigma_{f/g,l}^2 = 0$, $\forall l = 1,\ldots,d$;
2. step size $a_k = \frac{1}{10+k}$ and perturbation $c_k = (1+k)^{-1/3}$;
3. equal weights on response and gradient measurements, i.e., $\alpha_0 = 1/(d+1)$ and $W = I/(d+1)$;
4. perturbation vector for DiGARSM is taken from $\{[\pm 1, \pm 1, \ldots, \pm 1]\}$, i.e., $t_l = 1$ for $l = 1,2,\ldots,d$, and for SP-DiGARSM is sampled from a symmetric Bernoulli distribution that takes value ± 1 with probability 0.5, i.e., $\Delta_{k,l} \sim Ber(0.5)$ for $l = 1,2,\ldots,d$;
5. dimension is 4 ($d = 4$, so $x^* = [4,6,6,4]^T$), and each candidate point is sampled 3

times ($n = 3$);

6. starting point x_0 is generated randomly and fixed for each experiment, and each dimension of x_0 is sampled independently with $x_{0,l} \sim U[0,30]$ $l = 1,2,3,4$; and

7. each experiment is replicated independently 5 times, after which we plot the average relative squared 2-norm error (i.e., Mean Squared Error, MSE) between the optimum and each iterate to show the convergence speed.

We first evaluate the power of the additional gradient measurements in DiGARSM. Then we compare DiGARSM with SP-DiGARSM to explore the efficiency of SP method in high-dimensional problems. Finally, we evaluate the optimal weights against equal weights.

2.6.1 Efficiency with Additional Direct Gradient Estimate

In this part, we compare DiGARSM, defined in Equation (2.8), with its standard form, Equation (2.3), under the default settings. The results shown in Figure 2.1 demonstrate that with additional gradient information, SA with DiGARSM achieves a faster and smoother convergence. The error for SA with RSM increases for some iterations, which shows that the gradient estimate provided by RSM is not reliable, and it may provide an incorrect search direction.

2.6.2 Efficiency of SP-DiGARSM

In this part, we compare SP-DiGARSM and DiGARSM gradient estimators under default conditions. Two experiments were designed to show the efficiency of SP-

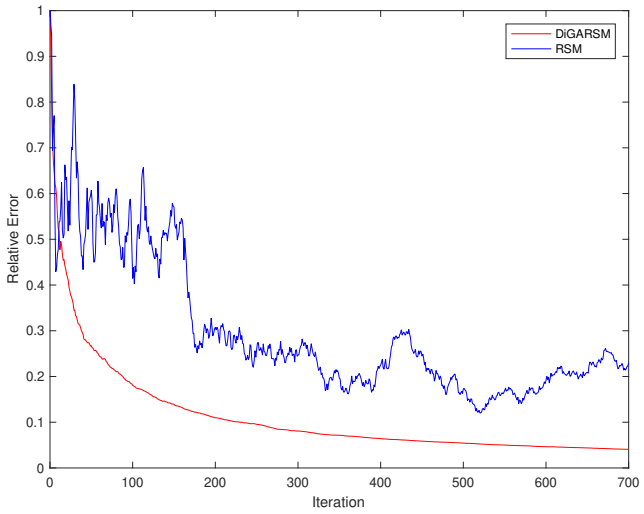


Figure 2.1: Average error of DiGARSM and RSM

DiGARSM. First, we fix the number of iterations for both algorithms. The results are shown in Figure 2.2.

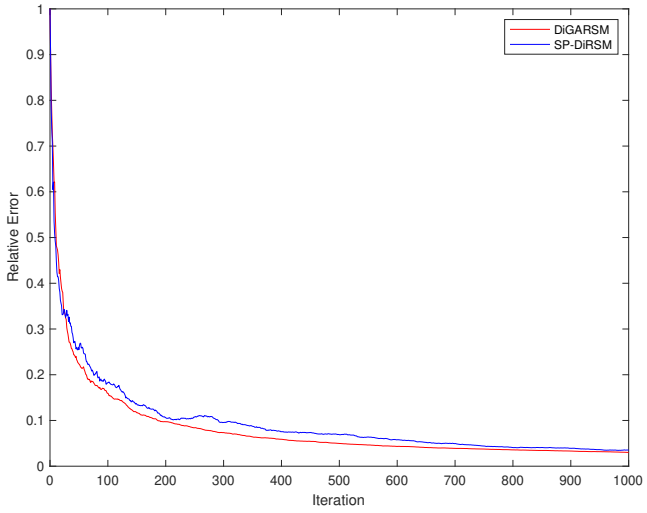


Figure 2.2: Average error of DiGARSM and SP-DiGARSM as a function of iterations

Since in each iteration, DiGARSM samples exponentially more than SP-DiGARSM, DiGARSM provides a better gradient estimate and results in a slightly faster convergences as shown in Figure 2.2. However, this invokes more computational and sampling cost, so

we fix the number of samples for DiGARSM and SP-DiGARSM in the second experiment (therefore, more iterations for SP-DiGARSM with the same number of samples). The results are shown in Figure 2.3, which shows that under the same computational and measurement budget, SP-DiGARSM exhibits a faster convergence than DiGARSM.

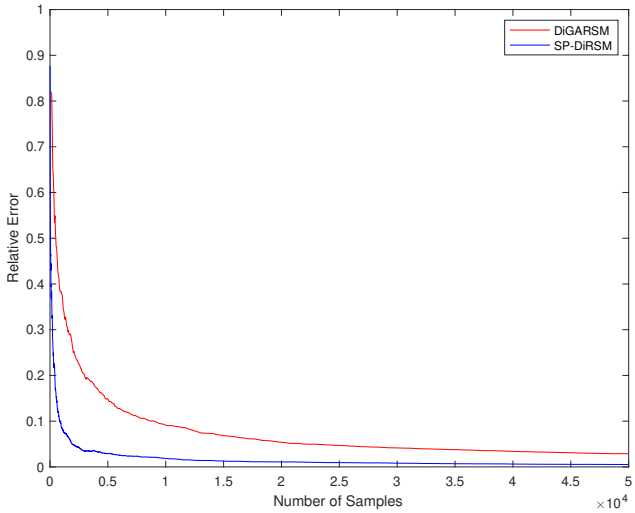


Figure 2.3: Average error of DiGARSM and SP-DiGARSM as a function of samples

2.6.3 Optimal Weighting

In this part, we evaluate the effect of weights on the convergence of DiGARSM SA algorithm. We assume the variance for the response and each dimension of gradient measurements are different, i.e., we set $\sigma_f^2 = 150$ and $\sigma_{g,l}^2 = l,\ l = 1,2,3,4$. We compare two DiGARSM algorithms with equal and optimal weights defined in Proposition 2.1. We consider two different settings: measurement variances known and unknown. In the first scenario, the optimal weights are calculated directly, whereas in the second case, the variances are estimated by the sample variances, then the weights are calculated. The results are shown in Figures 2.4 and 2.5, which both show the advantage of DiGARSM

with optimal weights.

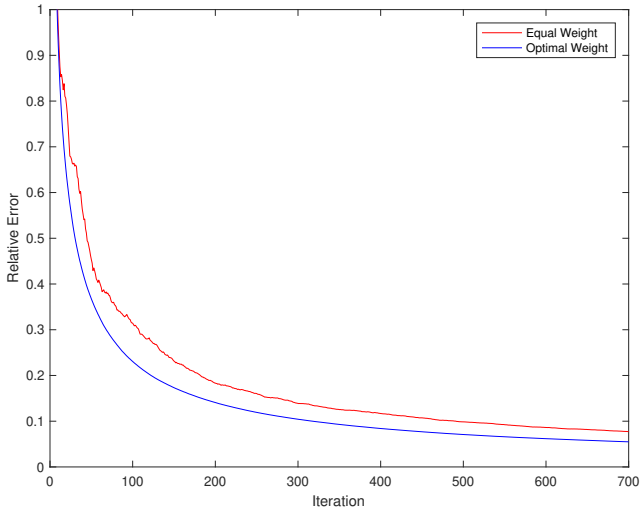


Figure 2.4: Average error of DiGARSM with equal and optimal weights: known variances

2.7 Conclusions and Future Research

In this chapter, we introduced a new stochastic approximation algorithm, DiGARSM, that utilizes both response and gradient measurements, combined through response surface methodology. The optimal weighting that minimizes the gradient estimate variance is proposed as a guideline for weight tuning. To address the high computational and sampling costs in high-dimensional problems, a revised algorithm with simultaneous perturbation, SP-DiGARSM, is presented. Under mild assumptions, convergence of DiGARSM and SP-DiGARSM is established. Finally, we demonstrated the efficiency of the proposed algorithms in simulation experiments.

In this work, we considered fixed sampling rate for each iteration. One possible way to further improve the performance is to consider dynamic sampling rate (i.e., denote the

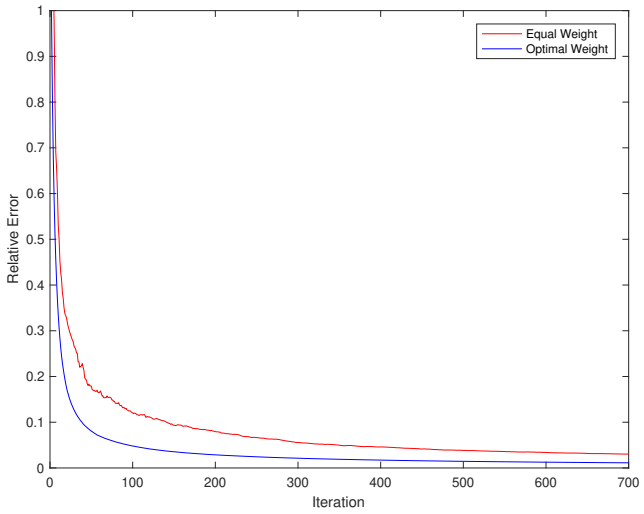


Figure 2.5: Average error of DiGARSM with equal and optimal weights: unknown variances

number of replications at iteration k by n_k, which grows as $k \to \infty$), with which a different convergence rate based on the sampling rate could be established [41]. In addition, if some prior information about the structure of the objective function is known, asymmetrical perturbation for the SP design could be investigated to improve the efficiency.

Chapter 3: Multi-Stage Stochastic Optimization

3.1 Introduction

In this chapter, we consider a reinforcement learning problem where an agent interacts with an underlying environment. A Markov Decision Process (MDP) with finite horizon is used to model the environment. In each move, the agent will take an action, receive a reward and land in a new state. The reward is usually random, and its distribution depends on both the state of the agent and the action taken. The distribution of the next state is also determined by the agent's current state and action. Our goal is to determine the optimal sequence of actions that leads to the highest expected reward. The optimality of the decision policy will be evaluated by the probability of correctly selecting the best action in the first stage of the underlying MDP.

If the distributions and the dynamics of the environment are known, the optimal set of actions can be computed through dynamic programming [42]. Under more general settings where the agent does not have perfect information regarding the environment, [43] proposed an adaptive algorithm based on a Multi-Armed Bandit (MAB) model and Upper Confidence Bound (UCB) [44]. [45] and [46] applied UCB to tree search, and [46] invented the term Monte Carlo Tree Search (MCTS) and used it in a Go-playing program for the first time. Since then, MCTS has been developed extensively and applied

to various games such as Othello [47] and Go [48]. To deal with different types of problems, several variations of MCTS have been introduced, e.g., Flat UCB (and its extension Bandit Algorithm for Smooth Trees) [49] and Single-Player MCTS (for single-player games) [50].

However, most bandit-based MCTS algorithms are designed to minimize regret (or maximize the cumulative reward of the agent), whereas in many situations, the goal of the agent may be to efficiently determine the optimal set of actions within a limited sampling budget. To the best of our knowledge, there is limited effort in the literature that aims at addressing the latter problem. [51] first incorporated Best Arm Identification (BAI) into MCTS for a MIN-MAX game tree, and provided upper bounds of play-outs under different settings. [52] had an objective similar to [51], but with a tighter bound. Their tree selection policy selects the node with largest confidence interval, which can be seen as choosing the node with the highest variance. In some sense, this is a pure exploration policy and would not efficiently use the limited sampling budget. In our work, we are motivated to establish a tree policy that intelligently balances exploration and exploitation (analogous to the objective of UCB). The algorithms developed in [51] and [52] are only for MIN-MAX game trees, whereas our new tree policy can be applied to more general types of tree search problems. The MCTS algorithm in [53] is more general than [51] and [52], but its goal is to estimate the maximum expected cumulative reward at the root node, whereas we focus on identifying the optimal action.

Algorithms that focus on minimizing regret tend to discourage exploration. This tendency can be seen in two ways. Suppose at some point an action was performed and received a small reward. To minimize regret, the algorithm would be discouraged from

taking this action again. However, the small reward could be due to the randomness in the reward distribution. Mathematically, [54] showed that for MAB algorithms, the number of times the optimal action is taken is exponentially more than sub-optimal ones, which makes sense when the objective is to maximize the cumulative reward, since the exploration of other actions is highly discouraged. This leads to our second motivation: is there a tree policy that explores sub-optimal actions more to ensure the optimal action is found?

Apart from the lack of exploration as a result of the underlying MAB model's objective to minimize regret or maximize cumulative reward, most MCTS algorithms assume that the support of the reward distribution is bounded and known (typically assumed to be $[0, 1]$). With the support of reward distribution being known, the parameter in the upper confidence term in UCB is tuned or the reward is normalized. However, a general tree search problem may likely have an unknown and practically unbounded range of rewards. In such case, assuming a range can lead to very poor performance. Therefore, the third motivation of our research is to relax the known reward support assumption.

To tackle the challenge in balancing exploration and exploitation with a limited sampling budget for a tree policy, we model the tree selection problem at each stage as a statistical *Ranking & Selection* (R&S) problem and propose a new tree policy for MCTS based on an adaptive algorithm from the R&S community. Similar to the MAB problem, R&S assumes that we are given a set of bandit machines (often referred to as alternatives in the R&S literature) with unknown reward distributions, and the goal is to select the machine with the highest mean reward. Specifically, we will develop an MCTS tree policy based on the Optimal Computing Budget Allocation (OCBA) framework [55].

OCBA was first proposed in [56], and aims at maximizing the probability of correctly selecting the action with highest mean reward using limited sampling budget. More recent developments of OCBA include addressing multiple objectives [57] and subset selection [58, 59].

The objective of the proposed OCBA tree policy is to maximize the Approximate Probability of Correct Selection (APCS), which is a lower bound on the probability of correctly selecting the optimal action at each node. Intuitively, the objective function of the new OCBA tree selection policy would lead to an optimal balance between exploration and exploitation with a limited sampling budget, and thus help address the drawbacks of existing work that either pursues pure exploration [51, 52] or exponentially discourages exploration [54]. Our new OCBA tree policy also removes the known and bounded support assumption for the reward distribution, because the new OCBA policy determines the sampling allocation based on the posterior distribution of each action, which is updated adaptively according to samples.

To summarize, contributions of this research include the following:

1. We propose a new tree policy for MCTS with an objective to maximize APCS with a limited sampling budget. The new tree policy optimally balances exploration and exploitation to efficiently select the optimal action. The new OCBA tree selection policy also relaxes the assumption of known bounded support on the reward distribution.

2. We present a sequential algorithm to implement the new OCBA tree policy that maximizes the APCS at each sampling stage and prove that our algorithm converges

to the optimal action.

3. We provide analyses on the convergence and the exploration-exploitation trade-off of the proposed algorithm, which works differently than bandit-based algorithms, and is more suitable for identifying the best action.

4. We demonstrate the efficiency of our algorithm through numerical experiments.

Remark 3.1. *In much of the computer science/artificial intelligence literature, an algorithm that focuses on determining the optimal set of actions under a limited budget is defined as a pure exploration algorithm (see, e.g., [60–62]), whereas we view such algorithms as retaining a balance between exploration and exploitation, as the analysis in Section 3.3 shows. In statistical R&S, pure exploration algorithms generally implies sampling based primarily on the variance of each action, which often leads to sampling suboptimal actions more. It will be clearer in the Section 3.5 where we show that OCBA-MCTS actually samples less those highly suboptimal actions and "exploits" those potential actions more.*

The rest of the research is organized as follows. We present the problem formulation in Section 3.2, and review the proposed OCBA-MCTS algorithm in Section 3.3. Theoretical analyses, including convergence theorems and exploration-exploitation analysis, are carried out in Section 3.4. Numerical examples are presented in Section 3.5 to evaluate the performance of our algorithm. Section 3.6 concludes the research and points to future research directions.

3.2 Problem Formulation

Consider a finite horizon MDP $M = (X, A, P, R)$ with horizon length H, finite state space X, finite action space A with $|A| > 1$, bounded reward function $R = \{R_t, t = 0, 1, \ldots H\}$ such that R_t maps a state-action pair to a random variable (r.v.), and transition function $P = \{P_t, t = 0, 1, \ldots H\}$ such that P_t maps a state-action pair to a probability distribution over X. We assume that P_t is unknown and/or $|X|$ and $|A|$ are very large, and hence it is not feasible to solve the problem by dynamic programming. Further define X_a and A_x as the available child states when taking action a and available actions at state x, respectively. Denote by $P_t(x,a)(y)$ the probability of transitioning to state $y \in X_a$ from state $x \in X$ when taking action $a \in A_x$ in stage t, and $R_t(x,a)$ the random reward in stage t by taking action a in state x. Let Π be the set of all possible nonstationary Markovian policies $\pi = \{\pi_i | \pi_i : X \to A, i \geq 0\}$.

Bandit-based algorithms for MDPs seek to minimize the expected cumulative regret, whereas our objective is to identify the best action that leads to maximum total expected reward given by $\mathbb{E}\big[\sum_{t=0}^{H-1} R_t(x_t, \pi_t(x_t))\big]$ for given $x_0 \in X$. We first define the optimal reward-to-go value function for state x in stage i by

$$V_i^*(x) = \max_{\pi \in \Pi} \mathbb{E}\big[\sum_{t=i}^{H-1} R_t(x_t, \pi_t(x_t)) \big| x_i = x\big],\ i = 0, 1, \ldots, H-1 \tag{3.1}$$

with $V_H^*(x) = 0$ for all $x \in X$. Also define

$$Q_i(x,a) = \mathbb{E}[R(x,a)] + \sum_{y \in X_a} P_t(x,a)(y) V_{i+1}^*(y),$$

with $Q_H(x,a) = 0$. It is well known [42] that eq. (3.1) can be written via the standard Bellman optimality equation:

$$\begin{aligned} V_i^*(x) &= \max_{a \in A_x} (\mathbb{E}[R_i(x,a)] + \mathbb{E}_{P_i(x,a)} V_{i+1}^*(Y)), \\ &= \max_{a \in A_x} (\mathbb{E}[R_i(x,a)] + \sum_{y \in X_a} P_i(x,a)(y) V_{i+1}^*(y)) \\ &= \max_{a \in A_x} (Q_i(x,a)), \ i = 0, 1, \ldots, H-1, \end{aligned}$$

where $Y \sim P_i(x,a)(\cdot)$ represents the random next state.

Since we are considering a tree search problem, some additional notation and definitions beyond MDP settings are needed. Define a state node by a tuple that contains the state and the stage number:

$$\mathbf{x} = (x,i) \in \mathbf{X}, \ \forall x \in X, \ 0 \leq i \leq H,$$

where $\mathbf{X}$ is the set of state nodes. Similarly, we define a state-action node by a tuple of state, stage number and action (i.e., a state node followed by an action):

$$\mathbf{a} = (\mathbf{x}, a) = (x,i,a), \ \forall x \in X, \ 0 \leq i \leq H, \ a \in A_x,$$

Now, we can rewrite, immediate reward function, value function for state, state-action pair with state node and state-action node and state transition distribution, respec-

tively, by

$$R(\mathbf{a}) = R(\mathbf{x}, a) := R_i(x, a)$$

$$V^*(\mathbf{x}) := V_i^*(x),$$

$$Q(\mathbf{a}) = Q(\mathbf{x}, a) := Q_i(x, a)$$

$$P(\mathbf{a}) = P(\mathbf{x}, a) := P_i(x, a).$$

Similarly, $V^*(\mathbf{x})$ and $Q(\mathbf{x}, a)$ are assumed to be zero for all terminal state nodes $\mathbf{x}$. To make our presentation clearer, we adopt the following definitions based on nodes: define $N(\mathbf{x})$ and $N(\mathbf{x}, a)$ the number of visits to node $\mathbf{x}$ and $(\mathbf{x}, a)$, respectively, $\mathbf{X_a}$ the set of child state nodes given parent nodes, and $A_\mathbf{x}$ the set of available child actions at node $\mathbf{x}$, respectively.

Traditionally, MCTS algorithms aim at estimating $V^*(\mathbf{x})$ and model the selection process in each stage as an MAB problem, i.e., view $Q(\mathbf{x}, a)$ as a set of bandit machines where $(\mathbf{x}, a)$ are child state-action nodes of $\mathbf{x}$ ([43, 45]), and minimize the *regret*, namely,

$$\begin{aligned} &\min_{a_1,\ldots,a_N \in A_\mathbf{x}} \{N \max_{a \in A_\mathbf{x}} (Q(\mathbf{x}, a)) - \sum_{k=1}^{N} Q(\mathbf{x}, a_k)\} \\ &= \{NV^*(\mathbf{x}) - \sum_{k=1}^{N} Q(\mathbf{x}, a_k)\} \end{aligned}$$

for $\mathbf{x}$ in stage $1, 2, \ldots H$, where N and a_k are the number of rollouts/simulations (also known as total sampling budget in much of Ranking & Selection literature) and the k-th action sampled at state node $\mathbf{x}$ by the tree policy, respectively. The meaning of rollout will be clearer in Section 3.3. In this research, our goal is to identify the optimal action

that achieves the highest cumulative reward at the root with initial state x, that is, find

$$a^*_{\mathbf{x}_0} = \arg\max_{a \in A_{\mathbf{x}_0}} Q(\mathbf{x}_0, a),$$

where the root state node $\mathbf{x}_0 = (x, 0)$. Let $\hat{Q}(\mathbf{x}, a) = R(\mathbf{x}, a) + V^*(\mathbf{y})$ be the random cumulative reward by taking action a at state node $\mathbf{x}$, where $\mathbf{y}$ is the random state node reached. Clearly, $\hat{Q}(\mathbf{x}, a)$ is a random variable. We assume $\hat{Q}(\mathbf{x}, a)$ is normally distributed with known variance, and its mean $\mu(\mathbf{x}, a)$ has a conjugate normal prior with a mean equals $Q(\mathbf{x}, a)$. Hence we have

$$Q(\mathbf{x}, a) = \mathbb{E}[\mathbb{E}[\hat{Q}(\mathbf{x}, a) | \mu(\mathbf{x}, a)]].$$

Remark 3.2. *For our derivations, we assume the variance of the sampling distribution of $\hat{Q}(x, a)$ is known; however, in practice, this prior variance may be unknown, in which case estimates such as the sample variance are used [63].*

Consider the non-informative case, i.e., the prior mean $Q(\mathbf{x}, a)$ is unknown, it can be shown that [64] the posterior of $\mu(\mathbf{x}, a)$ given observations (i.e., samples) is also normal. For convenience, define the t-th sample by $\hat{Q}^t(\mathbf{x}, a)$. Then the conditional distribution of $\mu(\mathbf{x}, a)$ given the set of samples $(\hat{Q}^1(\mathbf{x}, a), \hat{Q}^2(\mathbf{x}, a), \ldots, \hat{Q}^{N(\mathbf{x}, a)}(\mathbf{x}, a))$ is

$$\tilde{Q}(\mathbf{x}, a) \sim \mathcal{N}(\bar{Q}(\mathbf{x}, a), \frac{\sigma^2(\mathbf{x}, a)}{N(\mathbf{x}, a)}), \tag{3.2}$$

where

$$\bar{Q}(\mathbf{x},a) = \frac{1}{N(\mathbf{x},a)} \sum_{t=1}^{N(\mathbf{x},a)} \hat{Q}^t(\mathbf{x},a),$$

$$\tilde{Q}(\mathbf{x},a) = \mu(\mathbf{x},a)|(\hat{Q}^1(\mathbf{x},a), \hat{Q}^2(\mathbf{x},a), \ldots, \hat{Q}^{N(x,a)}(\mathbf{x},a)),$$

and $\sigma^2(\mathbf{x},a)$ is the variance of $\hat{Q}(\mathbf{x},a)$ and can be approximated by the sample variance:

$$\hat{\sigma}^2(\mathbf{x},a) = \frac{1}{N(\mathbf{x},a)} \sum_{t=1}^{N(\mathbf{x},a)} \left(\hat{Q}^t(\mathbf{x},a) - \bar{Q}(\mathbf{x},a)\right)^2.$$

Remark 3.3. *If the samples of $Q(\mathbf{x},a)$ are not normally distributed, the normal assumption can be justified by batch sampling and the central limit theorem.*

Under these settings, our objective is to maximize the Probability of Correct Selection (PCS) defined by

$$PCS = P\left[\bigcap_{a\in A, a\neq \hat{a}^*_{\mathbf{x}}} (\tilde{Q}(\mathbf{x},\hat{a}^*_{\mathbf{x}}) \geq \tilde{Q}(\mathbf{x},a))\right] \tag{3.3}$$

for a state node $\mathbf{x}$, where $\hat{a}^*_{\mathbf{x}}$ is the action that achieves the highest mean sample Q-value at such node, i.e., $\hat{a}^*_{\mathbf{x}} = \arg\max_{a\in A_{\mathbf{x}}} \bar{Q}(\mathbf{x},a)$.

PCS is hard to compute because of the intersections in the (joint) probability. We seek to simplify the joint probability by changing the intersections to sums using the Bonferroni inequality to make the problem tractable. By the Bonferroni inequality, PCS

is lower bounded by the Approximate Probability of Correct Selection (APCS), that is,

$$PCS \geq 1 - \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} P\left[\tilde{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) \leq \tilde{Q}(\mathbf{x}, a)\right] \tag{3.4}$$

$$=: APCS.$$

The objective of our new tree policy is to maximize APCS as given in Equation (3.4). Compared to MAB's objective of minimizing the expected cumulative regret, this objective function will result in an allocation of sampling budget to alternative actions in a way that optimally balances exploration and exploitation. This objective function is motivated by the OCBA algorithm [55] in the R&S literature. We will present and analyze our OCBA tree policy in the following sections.

3.3 Algorithm Description

In this section, we first briefly describe the main four phases, i.e., *selection*, *expansion*, *simulation* and *backpropagation*, in an MCTS algorithm. Then, we propose a novel tree policy in the selection stage that aims at finding the optimal action at each state node.

3.3.1 Canonical MCTS Algorithm

Here we briefly summarize the four phases in a typical MCTS algorithm. We refer readers to [65] for a complete illustration of these phases. Algorithm 3 represents a canonical MCTS, with detailed descriptions of the main phases below.

3.3.1.1 Selection

In this phase, the algorithm will navigate down the tree from the root state node to an expandable node, i.e., a node with unvisited child nodes. We assume that expansion is automatically followed when a state-action is encountered. Therefore, when determining the path down, there are three possible situations:

i If a state-action node is encountered (denoted by $(\mathbf{x}, a)$), we will land into a new state node $\mathbf{y}$ which is obtained by calling the expansion function. Then, we continue with the selection algorithm.

ii If an expandable state node (which could be a leaf node) is encountered, we call the expansion function to add a new child state-action node and a state node (by automatically expanding the state-action node) to the path. Then, we stop the selection phase and return the path from the root to this state node. Finally, we proceed with the simulation and backpropagation phase

iii If an unexpandable state node is encountered (denoted by $\mathbf{x}$), we employ a *tree policy* to determine which child action to sample. Then we enter the new state-action node $(\mathbf{x}, a)$ and continue the selection algorithm with this state-action node. The tree policies can be briefly categorized into two types: deterministic, such as UCB1 and several of its variants (e.g., UCB-tuned, UCB-E), and stochastic, such as ε-greedy and EXP3; see [65] for a review.

3.3.1.2 Expansion

In this phase, a random child state or state-action node of the given node is added. If the incoming node is a state node $\mathbf{x}$, the next node is selected randomly (usually uniform) from those unvisited child state-action nodes. If the incoming node is a state-action node $(\mathbf{x}, a)$, the subsequent state node is found by simply sampling from distribution $P(\mathbf{x}, a)(\cdot)$.

3.3.1.3 Simulation

In some literature, this phase is also known as "rollout". The simulation phase starts with a state node. The purpose of this step is to simulate a path from this node to a terminal node and produce a sample of cumulative reward by taking this path (which is a sample of the value for this node). The simulated path is taken by a *default policy*, which is usually sample the feasible child sate-action nodes uniformly. With this node's value sample, we may proceed to the backpropagation phase.

3.3.1.4 Backpropagation

This phase simply takes the simulated node value and update the values of the nodes in the path (obtained in selection step) backward.

In the next section, we will propose our tree policy based on OCBA and illustrate the detailed implementations of the four phases.

3.3.2 OCBA Selection Algorithm

We now present an efficient tree policy to estimate the optimal actions in every state node by estimating $V^*(\mathbf{x})$ and $Q(\mathbf{x}, a)$ for all possible $a \in A_{\mathbf{x}}$ at the state node. Denote the estimates of $V^*(\mathbf{x})$ at node $\mathbf{x}$ by $\hat{V}^*(\mathbf{x})$, which is initialized to 0 for all state nodes. Our algorithm estimates $Q(\mathbf{x}, a)$ for each action a by its sample mean, and selects the action that maximizes the sample mean as $\hat{a}^*_{\mathbf{x}}$. During the process, the estimate of $Q(\mathbf{x}, a)$ is given by Equation (3.2) and the proposed new OCBA tree policy is applied. Our algorithm follows the algorithmic framework described in Section 3.3.1, with the tree policy changed to OCBA and other mild modifications.

The structure of the proposed OCBA-MCTS algorithm is shown in Algorithms 3 to 8. There are two major characteristics: the first is to use the proposed OCBA algorithm for the tree policy. The second is to require each state-action node to be expanded $n_0 > 1$ times, because we need a sample variance for each state-action node, which will become clearer after the tree policy illustration. The process is run for a prespecified N times (which will be later referred to as number of rollouts or sampling budget) from the root state node $\mathbf{x}_0$, after which a partially expanded tree is obtained and the optimal action $\hat{a}^*_{\mathbf{x}_0}$ can be derived.

When steering down the tree and a state node $\mathbf{x}$ is visited, the selection phase, which is illustrated in Algorithm 4, will first determine if there is a child state-action node that was visited for less than n_0 times at the given state node. If there is, then the state-action node will be sampled and added to the path. In other words, we try to expand each state node when it is visited, and require each node to be expanded n_0 times. If all the state-

action nodes are well-expanded, Algorithm 4 will call Algorithm 5 (OCBASelection), which calculates the allocation of samples to child state-action nodes of the current state node for a total sampling budget $\sum_{a\in A} N(\mathbf{x},a)+1$. To determine the number of samples allocated to each state-action node, denoted by $(\tilde{N}(\mathbf{x},a_1),\tilde{N}(\mathbf{x},a_2),\ldots,\tilde{N}(\mathbf{x},a_{|A_\mathbf{x}|}))$ (where $a_i \in A_\mathbf{x}, i=1,\ldots,|A_\mathbf{x}|$), the OCBA tree policy first identifies the child state-action node with the largest sample mean (sample optimal) and finds the difference between the sample means of the sample optimum and all other nodes:

$$\hat{a}^*_\mathbf{x} := \arg\max_a \bar{Q}(\mathbf{x},a)$$

$$\delta_\mathbf{x}(\hat{a}^*_\mathbf{x},a) := \bar{Q}(\mathbf{x},\hat{a}^*_\mathbf{x}) - \bar{Q}(\mathbf{x},a),\ \forall a \neq \hat{a}^*_\mathbf{x}.$$

The set of allocations $(\tilde{N}(\mathbf{x},a_1),\tilde{N}(\mathbf{x},a_2),\ldots,\tilde{N}(\mathbf{x},a_{|A|}))$ that maximizes APCS can be obtained by solving the following set of equations:

$$\frac{\tilde{N}(\mathbf{x},a_{n+1})}{\tilde{N}(\mathbf{x},a_n)} = \left(\frac{\sigma(\mathbf{x},a_{n+1})/\delta_\mathbf{x}(\hat{a}^*_\mathbf{x},a_{n+1})}{\sigma(\mathbf{x},a_n)/\delta_\mathbf{x}(\hat{a}^*_\mathbf{x},a_n)}\right)^2,$$

$$\forall a_n, a_{n+1} \neq \hat{a}^*_\mathbf{x},\ a_n, a_{n+1} \in A_\mathbf{x}, \tag{3.5}$$

$$\tilde{N}(\mathbf{x},\hat{a}^*_\mathbf{x}) = \sigma(\mathbf{x},\hat{a}^*_\mathbf{x})\sqrt{\sum_{a\in A, a\neq\hat{a}^*_\mathbf{x}} \frac{(\tilde{N}(\mathbf{x},a))^2}{\sigma^2(\mathbf{x},a)}}, \tag{3.6}$$

$$\sum_{a\in A}\tilde{N}(\mathbf{x},a) = \sum_{a\in A} N(\mathbf{x},a)+1. \tag{3.7}$$

The derivations of Equations (3.5) to (3.7) are illustrated in the appendix.

After the new budget allocation is computed, the algorithm will select the "most

starving" action to sample [63], i.e., sample

$$\hat{a} = \arg\max_{a \in A_{\mathbf{x}}} (\tilde{N}(\mathbf{x}, a) - N(\mathbf{x}, a)). \tag{3.8}$$

We highlight some major modifications to the canonical MCTS in the proposed algorithm. First, in the selection phase, we will try to expand all "expandable" nodes visited when obtaining a path to leaf. Since the variances of the values of a state node's child nodes are required in the proposed tree policy, we define a state node as expandable if it has child nodes that are visited less than $n_0 > 1$ times. State-action nodes are always expandable.

At the expansion phase as shown in Algorithm 6, a state-action node is expanded by simply sampling the transition distribution $P(\mathbf{x}, a)(\cdot)$, and the resulting state node is subsequently added to the path. The reward by taking the action in the state node is also recorded and will be used in the backpropagation stage.

In the simulation and backpropagation phases illustrated in Algorithm 7 and 8, a leaf-to-terminal path is simulated, and its reward is used to update the value for the leaf node. If we denote the leaf node and the reward from the simulated path by $\mathbf{x}_l$ and r, respectively, the leaf node value estimate is updated by

$$\hat{V}^*(\mathbf{x}_l) \leftarrow \frac{N(\mathbf{x}_l) - 1}{N(\mathbf{x}_l)} \hat{V}^*(\mathbf{x}_l) + \frac{1}{N(\mathbf{x}_l)} r. \tag{3.9}$$

After updating the leaf state node, we update the nodes in the path collected in selection

stage in reversed order. Suppose we have a path

$$(\mathbf{x}_0, (\mathbf{x}_0, a_0), \ldots, \mathbf{x}_i, (\mathbf{x}_i, a_i), \mathbf{x}_{i+1}, \ldots, \mathbf{x}_l)$$

and the node values of $\mathbf{x}_{i+1}, \ldots, \mathbf{x}_l$ have been updated, the preceding nodes $\mathbf{x}_i$ and $(\mathbf{x}_i, a_i)$ are updated through

$$\hat{Q}^{N(\mathbf{x},a)}(\mathbf{x}_i, a_i) = R(\mathbf{x}_i, a) + \hat{V}^*(\mathbf{x}_{i+1}), \tag{3.10}$$

$$\bar{Q}(\mathbf{x}_i, a_i) \leftarrow \frac{N(\mathbf{x}_i, a_i) - 1}{N(\mathbf{x}_i, a_i)} \bar{Q}(\mathbf{x}_i, a_i) + \frac{1}{N(\mathbf{x}_i, a_i)} Q^{N(\mathbf{x},a)}(\mathbf{x}_i, a_i), \tag{3.11}$$

$$\bar{V}(\mathbf{x}_i) \leftarrow \frac{N(\mathbf{x}_i) - 1}{N(\mathbf{x}_i)} \bar{V}^*(\mathbf{x}_i) + \frac{1}{N(\mathbf{x}_i)} \bar{Q}(\mathbf{x}_i, a_i), \tag{3.12}$$

$$\hat{V}(\mathbf{x}_i) \leftarrow (1 - \alpha_{N(\mathbf{x}_i)}) \bar{V}(\mathbf{x}_i) + \alpha_{N(\mathbf{x}_i)} \max_{a \in A_{\mathbf{x}_i}} \bar{Q}(\mathbf{x}_i, a), \tag{3.13}$$

where $\bar{V}(\cdot)$ is an intermediate variable that records the average value of the node through the root-to-leaf path, and $\alpha_{N(\mathbf{x}_i)} \in [0, 1]$ is a smoothing parameter. The updates are performed backwards to the root node.

Details of the OCBA tree policy are shown in Algorithm 3 to 8.

Algorithm 3: MCTS

Input: Simulation budget (roll-out number) N, root state node $\mathbf{x}_0$
Output: $\hat{a}^*_{\mathbf{x}_0}$, $\hat{V}^*(\mathbf{x}_0)$

Set simulation counter $n \leftarrow 0$
while $n < N$ **do**
    $path \leftarrow selection(x_0)$
    $leaf \leftarrow path[end]$
    $r \leftarrow simulate(leaf)$
    $backpropagate(path, r)$
    $n \leftarrow n + 1$
return action $\hat{a}^*_{\mathbf{x}_0} = \arg\max_{a \in A} \bar{Q}(\mathbf{x}_0, a)$

Algorithm 4: *selection*$(\mathbf{x}_0)$

Input: root state node $\mathbf{x}_0$

Sample a root-to-leaf path.
$path \leftarrow (\ )$
$\mathbf{x} \leftarrow \mathbf{x}_0$
while *True* **do**
    Append state node $\mathbf{x}$ to *path*
    $N(\mathbf{x}) \leftarrow N(\mathbf{x}) + 1$
    if $\mathbf{x}$ *is a terminal node* **then**
        return *path*
    if $\mathbf{x}$ *is expandable* **then**
        $\hat{a} \leftarrow expand(\mathbf{x})$
        $\mathbf{y} \leftarrow expand((\mathbf{x}, \hat{a}))$
        Append state-action node $(\mathbf{x}, \hat{a})$ and leaf state node $\mathbf{y}$ to *path*
        $N(\mathbf{x}, a) \leftarrow N(\mathbf{x}, a) + 1$
        $N(\mathbf{x}) \leftarrow N(\mathbf{x}) + 1$
        return *path*
    else
        $\hat{a} \leftarrow OCBAselection(\mathbf{x})$
        Append state-action node $(\mathbf{x}, \hat{a})$ to *path*
        $N(\mathbf{x}, \hat{a}) \leftarrow N(\mathbf{x}, a) + 1$
        $\mathbf{x} \leftarrow expand((\mathbf{x}, \hat{a}))$

Algorithm 5: *OCBASelection*$(\mathbf{x})$

Input: state node $\mathbf{x}$

Identify $\hat{a}^*_{\mathbf{x}} = \arg\max_a \bar{Q}(\mathbf{x}, a)$
$\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, a) \leftarrow \bar{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) - \bar{Q}(\mathbf{x}, a)$
Compute new sampling allocation $(\tilde{N}(\mathbf{x}, a_1), \tilde{N}(\mathbf{x}, a_2), \ldots, \tilde{N}(\mathbf{x}, a_{|A|}))$
by solving Equations (3.5) to (3.7)
$\hat{a} \leftarrow \arg\max_{a \in A}(\tilde{N}(\mathbf{x}, a) - N(\mathbf{x}, a))$
return $\hat{a}$

Algorithm 6: *expand*$(\mathbf{x}$ *or* $(\mathbf{x},a))$

Input: a state node $\mathbf{x}$ *or* a state-action node $(\mathbf{x},a)$
Output: child node to be added to the tree

if *the input node is a state node* $\mathbf{x}$ **then**
    $S \leftarrow \{$feasible actions of state x that has been sampled less than n_0 times$\}$
    $\hat{a} \leftarrow$ random choice of S
    Add $(\mathbf{x},\hat{a})$ to the tree if it is unvisited
    return $\hat{a}$
else
    Sample node $(\mathbf{x},a)$ at state node $\mathbf{x}$ and obtain the child state node $\mathbf{y} \sim P(\mathbf{x},a)(\cdot)$
    Add $\mathbf{y}$ to the tree if it is unvisited
    return $\mathbf{y}$.

Algorithm 7: *simulate*$(\mathbf{x})$

Input: state node $\mathbf{x}$
$r \leftarrow 0$
while *True* **do**
    if $\mathbf{x}$ *is not terminal* **then**
        find a random child state-action node $(\mathbf{x},a)$ of $\mathbf{x}$
        $r \leftarrow r + R(\mathbf{x},a)$
        sample a and obtain the child state node $\mathbf{y} \sim P(\mathbf{x},a)(\cdot)$
        $\mathbf{x} \leftarrow \mathbf{y}$
    else
        return r

Algorithm 8: *backpropagate*$(path, reward)$

Input: path to a leaf node *path*, simulated reward *reward*
for *node in reversed(path)* **do**
    Update node values through Equations (3.9) to (3.13).

There are a few points worth emphasizing in Algorithm 5. First, $\tilde{N}(\mathbf{x}, a_i)$ is the total number of samples for each action i after the allocation. Given present information, i.e., all samples state node $\mathbf{x}$, OCBA-MCTS assumes now a total number of $\sum_{a \in A} N(\mathbf{x}, a) + 1$ samples available. By solving Equations (3.5) to (3.7), the new budget allocation $(\tilde{N}(\mathbf{x}, a_1), \tilde{N}(\mathbf{x}, a_2), \ldots, \tilde{N}(\mathbf{x}, a_{|A|}))$ that maximizes APCS is calculated. Afterwards, one action based on Equation (3.8) is selected to sample and move to the next stage. This "most-starving" implementation of the OCBA policy as given in Algorithm 5 is fully sequential, as each iteration allocates only one sample to an action before the allocation decision is recomputed. It is also possible to allocate the sampling budget in a batch of size $\Delta > 1$. We use the "most-starving" scheme, because it has been shown to be more efficient than the batch sampling scheme [66]. However, the benefit of sampling in batches for MCTS is that in one iteration, multiple root-to-leaf paths can be examined, enabling parallelization of the algorithm. We will consider this in future research.

Second, updating $\hat{V}(\mathbf{x}_i)$ involves two stages: updating the value estimate along the path (Equation (3.12)) and taking the maximum over the values of the child state-action nodes (canonical way to update). Then the two values are mixed through $\alpha_{N(\mathbf{x}_i)}$ to update $\hat{V}(\mathbf{x}_i)$, as prior research (e.g., [46, 67]) suggests mixing with $\alpha_{N(\mathbf{x}_i)} \to 1$ (i.e., asymptotically achieves Bellman update) ensures more stable updates.

Finally, although we present our algorithm in the context of solving an MDP, it can be applied to other tree structures such as MIN-MAX game trees or more general game trees, by setting the reward function and the max and min operators accordingly.

3.4 Analysis of OCBA-MCTS

In this section, we first analyze how the OCBA tree policy in OCBA-MCTS balances exploration and exploitation mathematically. Then, we present several theoretical results regarding OCBA-MCTS. The proofs are given in the appendix.

3.4.1 Exploration-Exploitation Balance

Equations (3.5) to (3.7) determine the new sampling budget allocation. First, eq. (3.5) shows that the sub-optimal state-action nodes should be sampled proportional to their variances and inversely proportional to the squared differences between their sample means and that of the optimal state-action node. This represents a different type of trade off between exploration (sampling actions with high variances) and exploitation (sampling actions with higher sample means) compared to bandit-based algorithms.

3.4.2 Convergence Analysis

In this part, we present three theorems regarding OCBA-MCTS. The first theorem ensures the estimate of the value-to-go function converges to the true value. The second theorem proves that OCBA-MCTS will select the correct action, i.e., the PCS converges to 1. The last theorem guarantees that the APCS, which is a lower bound of PCS, is maximized by solving Equations (3.5) and (3.6) in each step. It is shown that at each point of the tree policy when a decision needs to be made, the action that maximizes the APCS will be selected and sampled. Therefore, the OCBA tree policy gradually maximizes the overall APCS at the root, which is a lower bound for PCS.

To prove that our algorithm correctly selects the optimal action as the sampling budget goes to infinity, we first prove that at each stage, the PCS converges to 1. The process of our algorithm at each single stage is OCBA adapted from [55]. OCBA tries to identify the alternative with highest mean from a set of normal random variables (alternatives) with means J_i and known variances σ_i^2, $i = 1,2,\ldots,k$ by efficiently allocating samples that maximizes APCS. OCBA assumes that J_i is also normally distributed. Here we present OCBA again in Algorithm 9 for convenience. The budget allocation process is similar to Equations (3.5) to (3.7). First define

$$\bar{J}_i := \frac{1}{l_i}\sum_{m=1}^{l_i} \hat{J}_i^m,$$

$$b := \arg\max_i \bar{J}_i,$$

$$\delta(b,i) := \bar{J}_b - \bar{J}_i,\ \forall i \neq b,$$

where l_i is the number of samples for alternative i, $\hat{J}_i^m$ is the m-th sample of J_i for $1 \leq i \leq k$, $1 \leq m \leq l_i$. The new allocations $(\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_k)$ with budget $T > \sum_i l_i$ can be obtained by solving the set of equations:

$$\frac{\tilde{l}_i}{\tilde{l}_j} = \left(\frac{\sigma_i/\delta(b,i)}{\sigma_j/\delta(b,j)}\right)^2,\ \forall i \neq j \neq b, \tag{3.14}$$

$$\tilde{l}_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{\tilde{l}_i^2}{(\sigma_i)^2}}, \tag{3.15}$$

$$\sum_{i=1} \tilde{l}_i = T, \tag{3.16}$$

where σ_i is the standard deviation of the $i-$th reward distribution. As in Remark 2, σ_i

is assumed to be known, but in practice can be unknown and approximated by sample standard deviation $\hat{\sigma}_i = \sqrt{\frac{1}{l_i}\sum_{m=1}^{l_i}(\hat{J}_i^m - \bar{J}_i)^2}$.

Algorithm 9: One-stage OCBA

Input: Total sampling budget T, initial sample size n_0
Output: Index of optimal action $\hat{b}$

Sample each of the k alternatives n_0 times;
Set counter $l_i \leftarrow n_0 \ \forall i = 1,2,\ldots,k$;
$l \leftarrow kn_0$;
Calculate $\bar{J}_i$ and $\hat{\sigma}_i^2$, $\forall i = 1,2,\ldots,k$;
while $l \leq T$ **do**
    Compute new budget allocation $(\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_k)$ by solving eq. (3.14)-(3.16) with budget $l+1$;
    Sample $\hat{i} = \arg\max_{1\leq i\leq k}(\tilde{l}_i - l_i)$;
    Update $\bar{J}_{\hat{i}}$ (and $\hat{\sigma}_{\hat{i}}^2$ if sample variance is used);
    $l_{\hat{i}} \leftarrow l_{\hat{i}} + 1$;
    $l \leftarrow l + 1$;
return $\hat{b} = \arg\max_{1\leq i\leq k} \bar{J}_i$;

Lemma 3.1. *Given a set of k normal random variables (actions) with mean J_i and variance σ_i^2, $i = 1,2,\ldots,k$, where J_i. are also normally distributed. Suppose OCBA is run with sampling budget T. Define the PCS*

$$PCS = P\left[\bigcap_{i=1,i\neq b}^{k} (\tilde{J}_b - \tilde{J}_i) \geq 0\right],$$

where $\tilde{J}_i$ is the posterior distribution of J_i given l_i samples $\forall i = 1,2,\ldots,k$. Then, $PCS \to 1$ as $T \to \infty$.

Proof. The *PCS* can be lower bounded by APCS, i.e., by the Bonferroni inequality

$$\begin{aligned} PCS =& P\Big[\bigcap_{i=1,i\neq b}^{k}(\tilde{J}_b-\tilde{J}_i)\geq 0\Big] \\ \geq& 1-\sum_{i=1,i\neq b}^{k}P\Big[\tilde{J}_b-\tilde{J}_i\leq 0\Big] \\ =& APCS. \end{aligned}$$

Thus, to prove that $PCS\to 1$, it suffices to prove $APCS\to 1$, i.e.,

$$\sum_{i=1,i\neq b}^{k}P\Big[(\tilde{J}_b-\tilde{J}_i)\leq 0\Big]\to 0 \text{ as } T\to\infty.$$

Based on the normality assumption, the posterior distribution is also normal, i.e., $\tilde{J}_i\sim N(\bar{J}_i,\sigma_i^2/l_i)$. Thus, $\tilde{J}_b-\tilde{J}_i\sim N(\bar{J}_b-\bar{J}_i,\sigma_b^2/l_b+\sigma_i^2/l_i)$. Therefore,

$$\sum_{i=1,i\neq b}^{k}P\Big[(\tilde{J}_b-\tilde{J}_i)\leq 0\Big]=\sum_{i=1,i\neq b}^{k}\Phi(-\frac{\bar{J}_b-\bar{J}_i}{\sqrt{\sigma_b^2/l_b+\sigma_i^2/l_i}}), \tag{3.17}$$

where Φ is the cdf of the standard normal distribution. Since

$$\sum_{i=1}^{k}l_i=T,$$

then when $T\to\infty$, at least one of the actions will be sampled infinitely many times, i.e., there exists an index i such that $l_i\to\infty$. Then there are two possible cases: $i\neq b$ and $i=b$.

Case 1: $i\neq b$

According to eq. (3.14),

$$l_j = \left(\frac{\sigma_j/\delta(b,j)}{\sigma_i/\delta(b,i)} l_i \right)^2, \ \forall j \neq i, j \neq b.$$

Since σ_i and $\delta(b,i)$ are bounded for all i, $l_j \to \infty$, $\forall j \neq b$.

Therefore, by eq. (3.15), $l_b \to \infty$. Thus, $l_i \to \infty$ for all $i = 1,2,\ldots,k$.

Case 2: $i = b$ According to (3.15),

$$l_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{l_i^2}{(\sigma_i)^2}} \to \infty.$$

Thus there exists an index $i \neq b$ such that $l_i \to \infty$. By a similar argument in Case 1, we can conclude that $l_i \to \infty$ for all $i = 1,2,\ldots,k$.

In either case, we have $l_i \to \infty$ for all $i = 1,2,\ldots,k$. Additionally, since $\bar{J}_b$ is defined to be the maximum of all $\bar{J}_i$, i.e., $\bar{J}_b - \bar{J}_i \geq 0$ for all $i \neq b$, eq. (3.17) becomes

$$\sum_{i=1, i\neq b}^{k} P\left[(\tilde{J}_b - \tilde{J}_i) \leq 0 \right] = \sum_{i=1, i\neq b}^{k} \Phi\left(-\frac{\bar{J}_b - \bar{J}_i}{\sqrt{\sigma_b^2/l_b + \sigma_i^2/l_i}} \right) \to 0$$

as desired. □

A corollary follows directly from the lemma.

Corollary 3.1. *Suppose one-stage OCBA is run with budget T. Then*

$$\bar{J}_i \to \mathbb{E}[J_i] \ w.p.\ 1 \ as \ T \to \infty, \ \forall i = 1,2,\ldots,k.$$

The proof is a simple application of the strong law of large numbers, since $l_b \to \infty$. With these results, we are ready to show the three main theorems.

Theorem 3.1 (Asymptotic consistency)**.** *Assume the expected cumulative reward at state-action node* $(\mathbf{x}, a)$ *is a normal random variable with mean* $\mu(\mathbf{x}, a)$ *and variance* $\sigma^2(\mathbf{x}, a) < \infty$*, i.e.,* $\hat{Q}(\mathbf{x}, a) \sim \mathcal{N}(\mu(\mathbf{x}, a), \sigma^2(\mathbf{x}, a))$ *for* $0 \leq i < H$*. Further assume* $\mu(\mathbf{x}, a)$ *is also normally distributed with unknown mean and known variance. Suppose the proposed OCBA-MCTS algorithm is run with a sampling budget N at root state node* $\mathbf{x}_0$*. Then at any subsequent nodes* $\mathbf{x}$*,*

$$\lim_{N\to\infty} \bar{Q}(\mathbf{x}, a) = \mathbb{E}[\hat{Q}(\mathbf{x}, a)] = Q(\mathbf{x}, a),$$

$$\lim_{N\to\infty} \hat{V}(\mathbf{x}) = V^*(\mathbf{x}),\ \forall\, \mathbf{x} \in \mathbf{X},\ (\mathbf{x}, a) \in \mathbf{X} \times A_{\mathbf{x}}.$$

Proof of Theorem 3.1. The result can be proved by induction.

First observe that since $N \to \infty$, each path is explored infinitely many times. Thus the number of samples in each stage also goes to infinity as $N \to \infty$.

Suppose at some point of the algorithm, all nodes are expanded. If the current state node $\mathbf{x}$ is at stage $H-1$ (i.e., it will transit into a terminal node in the next transition), running Algorithm 5 reduces to a single-stage problem, which is the same as OCBA in Algorithm 9. $\hat{Q}(\mathbf{x}, a)$ can be viewed as a set of alternatives for $a \in A$. From Corollary 3.1, it is straightforward that

$$\lim_{N\to\infty} \bar{Q}(\mathbf{x}, a) = Q(\mathbf{x}, a).$$

Therefore, since the reward function is bounded

$$\begin{aligned}\lim_{N\to\infty} \hat{V}(\mathbf{x}) &= \lim_{N\to\infty} \max_{a\in A_{\mathbf{x}}} \bar{Q}(\mathbf{x},a) \\ &= \max_{a\in A_{\mathbf{x}}} \lim_{N\to\infty} \bar{Q}(\mathbf{x},a) \\ &= V^*(\mathbf{x}).\end{aligned}$$

Now suppose that the statement is true for all child state nodes $\mathbf{y}$ of a state $\mathbf{x}$, i.e., $\hat{V}(\mathbf{y}) \to V^*(\mathbf{y})$ and $\mathbf{y}$ could be achieved from $\mathbf{x}$. Then for $\mathbf{x}$, the algorithm also reduces to single-stage OCBA. Thus from Corollary 3.1 again

$$\begin{aligned}\lim_{N\to\infty} \bar{Q}(\mathbf{x},a) &= \lim_{N(\mathbf{x},a)\to\infty} \bar{Q}(\mathbf{x},a) \\ &= \mathbb{E}[R(\mathbf{x},a)] + \mathbb{E}_{P(\mathbf{x},a)}[V^*(\mathbf{y})] \\ &= Q(\mathbf{x},a)\end{aligned}$$

for all child state-action pair $(\mathbf{x},a)$. It follows that

$$\lim_{N\to\infty} \hat{V}(\mathbf{x}) \to V^*(\mathbf{x}).$$

□

Theorem 3.2 (Asymptotic correctness)**.** *Under the same assumptions of Theorem 3.1, the*

PCS converges to 1 for any state node $\mathbf{x} \in \mathbf{X}$*, i.e.,*

$$P\left[\bigcap_{a\in A_{\mathbf{x}}, a\neq \hat{a}^*_{\mathbf{x}}} (\lim_{N\to\infty} \tilde{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) - \lim_{N\to\infty} \tilde{Q}(\mathbf{x}, a)) \geq 0\right] = 1,$$

$$\forall\, \mathbf{x} \in \mathbf{X},$$

where $\hat{a}^*_{\mathbf{x}} = \arg\max_{a\in A_{\mathbf{x}}} \bar{Q}(\mathbf{x}, a)$.

Theorem 3.2 is a direct result of Lemma 3.1.

Proof of Theorem 3.2. Since we assume $\hat{Q}(\mathbf{x}, a)$ is normally distributed with known variance, the posterior distribution of $\hat{Q}(\mathbf{x}, a)$, i.e., $\tilde{Q}(\mathbf{x}, a)$, is also a normal random variable. Then, it follows directly from Lemma 3.1 that

$$P\left[\bigcap_{a\in A_{\mathbf{x}}, a\neq \hat{a}^*_{\mathbf{x}}}^{k} (\lim_{N\to\infty} \tilde{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) - \lim_{N\to\infty} \tilde{Q}(\mathbf{x}, a)) \geq 0\right] = 1,$$

$$\forall i = 1, \ldots, H,\ \mathbf{x} \in \mathbf{X},\ a \in A.$$

□

Theorem 3.3. *Under the same assumptions of Theorem 3.1, the APCS defined in Equation (3.4) is maximized asymptotically with simulation budget allocation* $(\tilde{N}(\mathbf{x}, a_1), \tilde{N}(\mathbf{x}, a_2), \ldots, \tilde{N}(\mathbf{x}, a_{|A_{\mathbf{x}}|}))$ *by solving Equations (3.5) and (3.6) with total budget N, i.e.,* $\sum_{a\in A_{\mathbf{x}}} \tilde{N}(\mathbf{x}, a) = N$.

Proof of Theorem 3.3. The problem of maximizing APCS with budget constraint can be

formulated as

$$\max_{\tilde{N}(\mathbf{x},a),a\in A_{\mathbf{x}}} 1-\sum_{a\in A,a\neq\hat{a}^*_{\mathbf{x}}} P\left[\tilde{Q}(\mathbf{x},\hat{a}^*_{\mathbf{x}})\leq\tilde{Q}(\mathbf{x},a)\right]$$

$$s.t.\sum_{a\in A_{\mathbf{x}}}\tilde{N}(\mathbf{x},a)=N.$$

With Lagrange multiplier λ, the Lagrangian can be written as

$$\begin{aligned}
L &= 1-\sum_{a\in A_{\mathbf{x}},a\neq\hat{a}^*_{\mathbf{x}}} P\left[\tilde{Q}(\mathbf{x},\hat{a}^*_{\mathbf{x}})\leq\tilde{Q}(\mathbf{x},a)\right]+\lambda(\sum_{a\in A_{\mathbf{x}}}\tilde{N}(\mathbf{x},a)-N)\\
&= 1-\sum_{a\in A_{\mathbf{x}},a\neq\hat{a}^*_{\mathbf{x}}}\Phi(\frac{\bar{Q}(\mathbf{x},a)-\bar{Q}(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{\sigma_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})})+\lambda(\sum_{a\in A_{\mathbf{x}}}\tilde{N}(\mathbf{x},a)-N)\\
&= 1-\sum_{a\in A_{\mathbf{x}},a\neq\hat{a}^*_{\mathbf{x}}}\Phi(-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{\sigma_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})})+\lambda(\sum_{a\in A_{\mathbf{x}}}\tilde{N}(\mathbf{x},a)-N),
\end{aligned}$$

where

$$\sigma^2_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})=\frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{N(\mathbf{x},\hat{a}^*_{\mathbf{x}})}+\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)}$$

Apply Karush-Kuhn-Tucker (KKT) conditions [68]:

- primal feasible

$$N(\mathbf{x},a)\geq 0,\forall a\in A \tag{3.18}$$

$$\sum_{a\in A_{\mathbf{x}}}\tilde{N}(\mathbf{x},a)-N=0, \tag{3.19}$$

- stationarity

$$\frac{\partial L}{\partial N(\mathbf{x},a)} = \frac{\partial L}{\partial(-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{\sigma_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})})} \frac{\partial(-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{\sigma_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})})}{\partial \sigma_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})} \frac{\partial \sigma_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})}{\partial N(\mathbf{x},a)}$$

$$= 0 \tag{3.20}$$

Case 1: $a \neq \hat{a}^*_{\mathbf{x}}$

$$\frac{\partial L}{\partial N(\mathbf{x},a)} = \frac{\sigma^2(\mathbf{x},a)\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{N^2(\mathbf{x},a)\sigma^3_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})\sqrt{2\pi}} \exp -\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\sigma^2_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})} + \lambda$$

$$= 0 \tag{3.21}$$

.

Case 2: $a = \hat{a}^*_{\mathbf{x}}$

$$\frac{\partial L}{\partial N(\mathbf{x},\hat{a}^*_{\mathbf{x}})} = \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} \frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{N^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})\sigma^3_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})\sqrt{2\pi}} \exp -\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\sigma^2_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})} + \lambda$$

$$= 0 \tag{3.22}$$

.

From Equation (3.21),

$$\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{\sigma^3_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})\sqrt{2\pi}} \exp -\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\sigma^2_{\mathbf{x}}(a,\hat{a}^*_{\mathbf{x}})} = -\lambda \frac{N^2(\mathbf{x},a)}{\sigma^2(\mathbf{x},a)}. \tag{3.23}$$

Plug Equation (3.23) into Equation (3.22) yields

$$\frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{N^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}\sum_{a\in A_{\mathbf{x}},a\neq\hat{a}^*_{\mathbf{x}}}\lambda\frac{N^2(\mathbf{x},a)}{\sigma^2(\mathbf{x},a)}=\lambda,$$

i.e.,

$$N(\mathbf{x},\hat{a}^*_{\mathbf{x}})=\sqrt{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})\sum_{a\in A_{\mathbf{x}},a\neq\hat{a}^*_{\mathbf{x}}}\frac{N^2(\mathbf{x},a)}{\sigma^2(\mathbf{x},a)}}. \tag{3.24}$$

After sufficiently large number of samples, we may conclude from Equation (3.24) that our algorithm would focus more on sampling the sample optimal. Thus, we may assume that $N(\mathbf{x},\hat{a}^*_{\mathbf{x}})\gg N(\mathbf{x},a)$ for all suboptimal actions $a\in A_{\mathbf{x}}$.

Now, for two suboptimal actions $a\neq\tilde{a}\neq\hat{a}^*_{\mathbf{x}}$, we have

$$\begin{aligned}&\frac{\sigma^2(\mathbf{x},a)\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{N^2(\mathbf{x},a)(\frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{N(\mathbf{x},\hat{a}^*_{\mathbf{x}})}+\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)})^{3/2}}\exp-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{2(\frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{N(\mathbf{x},\hat{a}^*_{\mathbf{x}})}+\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)})}\\&=\frac{\sigma^2(\mathbf{x},\tilde{a})\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})}{N\mathbf{x}^2(\mathbf{x},\tilde{a})(\frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{N\mathbf{x}(\mathbf{x},\hat{a}^*_{\mathbf{x}})}+\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})})^{3/2}}\exp-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})^2}{2(\frac{\sigma^2(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{N(\mathbf{x},\hat{a}^*_{\mathbf{x}})}+\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})})}.\end{aligned}$$

Apply the $N(\mathbf{x},\hat{a}^*_{\mathbf{x}})\gg N(\mathbf{x},a)$ assumption:

$$\begin{aligned}&\frac{\sigma^2(\mathbf{x},a)\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{N^2(\mathbf{x},a)(\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)})^{3/2}}\exp-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{2(\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)})}\\&=\frac{\sigma^2(\mathbf{x},\tilde{a})\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})}{N^2(\mathbf{x},\tilde{a})(\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})})^{3/2}}\exp-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})^2}{2(\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})})}.\end{aligned}$$

i.e.,

$$\left(\frac{N(\mathbf{x},\tilde{a})}{N(\mathbf{x},a)}\right)^{1/2} = \frac{\sigma^2(\mathbf{x},a)}{\sigma^2(\mathbf{x},\tilde{a})}\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})}{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}\exp\left(\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{2(\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)})} - \frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})^2}{2(\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})})}\right).$$

Taking logarithm on both sides yields

$$\log N(\mathbf{x},\tilde{a}) - \log(N(\mathbf{x},a)) = 2\log\frac{\sigma^2(\mathbf{x},a)}{\sigma^2(\mathbf{x},\tilde{a})}\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})}{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)} + \frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)}} - \frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})}}.$$

When the number of samples is sufficiently large ($N \to \infty$), the log terms can be neglected compared to the terms linear in $N(\mathbf{x},a)$ or $N(\mathbf{x},\tilde{a})$. Therefore, removing the log terms yields,

$$\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)}} = \frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)^2}{\frac{\sigma^2(\mathbf{x},\tilde{a})}{N(\mathbf{x},\tilde{a})}},$$

namely,

$$\frac{\tilde{N}(\mathbf{x},a)}{\tilde{N}(\mathbf{x},\tilde{a})} = \left(\frac{\sigma(\mathbf{x},a)/\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{\sigma(\mathbf{x},\tilde{a})/\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})}\right)^2,$$

$$\forall a\,,\tilde{a} \neq \hat{a}^*_{\mathbf{x}}.$$

□

Theorem 3.3, which follows from the result originally derived in [55], shows that at each point of the algorithm when a decision needs to be made, the action that maximizes

the APCS will be selected and sampled. Therefore, the OCBA tree policy gradually maximizes the overall APCS at the root, which is a lower bound for PCS.

3.4.3 Performance Lower Bound

We take advantage of the normal distribution assumptions on the Q functions and provide a lower bound on PCS.

Theorem 3.4 (Lower bound on the probability of correct selection)**.** *Under the same assumptions of Theorem 3.1, the PCS at each stage and state is lower bounded by*

$$PCS \geq 1 - \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} \Phi\left(- \frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, a)\sqrt{N(\mathbf{x}, \hat{a}^*_{\mathbf{x}})}}{\sqrt{\sigma^2(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) + \sigma(\mathbf{x}, \hat{a}^*_{\mathbf{x}})\sigma^2(\mathbf{x}, a)\sum_{\tilde{a} \in A_{\mathbf{x}}, \tilde{a} \neq \hat{a}^*_{\mathbf{x}}} \frac{r(\tilde{a}, a)}{\sigma(\mathbf{x}, \tilde{a})}}} \right),$$

where $\Phi(\cdot)$ is the cdf of standard normal distribution and

$$r_{\mathbf{x}}(\tilde{a}, a) = \frac{\sigma(\mathbf{x}, \tilde{a})\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, a)^2}{\sigma(\mathbf{x}, a)\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, \tilde{a})}.$$

Proof of Theorem 3.4. When the number of samples at node $\mathbf{x}$ is large, we assume that $N(\mathbf{x}, a)$ satisfies Equations (3.5) to (3.6).

From Equation (3.5), we have

$$N(\mathbf{x}, \tilde{a}) = \left(\frac{\sigma(\mathbf{x}, \tilde{a})\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, a)}{\sigma(\mathbf{x}, a)\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, \tilde{a})}\right)^2 N(\mathbf{x}, a), \quad \forall \tilde{a}, a \neq \hat{a}^*_{\mathbf{x}}. \tag{3.25}$$

In this way, we can express the budget allocation to any suboptimal action $\tilde{a}$ as the product

of the budget allocation to a particular suboptimal action a and the factor

$$r_{\mathbf{x}}(\tilde{a},a) = \Big(\frac{\sigma(\mathbf{x},\tilde{a})\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},a)}{\sigma(\mathbf{x},a)\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}},\tilde{a})}\Big)^2.$$

From Equation (3.6):

$$N(\mathbf{x},\hat{a}^*_{\mathbf{x}}) = \sigma(\mathbf{x},\hat{a}^*_{\mathbf{x}})\sqrt{\sum_{\tilde{a}\in A_{\mathbf{x}},\tilde{a}\neq\hat{a}^*_{\mathbf{x}}}\frac{(N(\mathbf{x},\tilde{a})^2)}{\sigma^2(\mathbf{x},\tilde{a})}}.$$

Substitute $N(\mathbf{x},\tilde{a})$ from Equation (3.25) yields

$$N(\mathbf{x},\hat{a}^*_{\mathbf{x}}) = N(\mathbf{x},a)\sigma(\mathbf{x},\hat{a}^*_{\mathbf{x}})\sqrt{\sum_{\tilde{a}\in A_{\mathbf{x}},\tilde{a}\neq\hat{a}^*_{\mathbf{x}}}\frac{(r_{\mathbf{x}}(\tilde{a},a))^2}{\sigma^2(\mathbf{x},\tilde{a})}},$$

i.e.,

$$N(\mathbf{x},a) = \frac{N(\mathbf{x},\hat{a}^*_{\mathbf{x}})}{\sigma(\mathbf{x},\hat{a}^*_{\mathbf{x}})\sqrt{\sum_{\tilde{a}\in A_{\mathbf{x}},\tilde{a}\neq\hat{a}^*_{\mathbf{x}}}\frac{(r_{\mathbf{x}}(\tilde{a},a))^2}{\sigma^2(\mathbf{x},\tilde{a})}}}.$$

Since PCS is lower bounded by APCS, and the posterior $\tilde{Q}(\mathbf{x},a)$ is normally distributed with

$$\tilde{Q}(\mathbf{x},a) \sim N(\bar{Q}(\mathbf{x},a), \frac{\sigma^2(\mathbf{x},a)}{N(\mathbf{x},a)}),$$

then

$$
\begin{aligned}
PCS &\geq APCS \\
&= 1 - \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} P\left[\tilde{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) \leq \tilde{Q}(\mathbf{x}, a)\right] \\
&= 1 - \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} \Phi\left(\frac{\bar{Q}(\mathbf{x}, a) - \bar{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}})}{\sigma_{\mathbf{x}}(a, \hat{a}^*_{\mathbf{x}})}\right) \\
&= 1 - \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} \Phi\left(-\frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, a)}{\sigma_{\mathbf{x}}(a, \hat{a}^*_{\mathbf{x}})}\right),
\end{aligned}
$$

where the second equality is because $\tilde{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) - \tilde{Q}(\mathbf{x}, a)$ is normally distributed with mean $\bar{Q}(\mathbf{x}, a) - \bar{Q}(\mathbf{x}, \hat{a}^*_{\mathbf{x}})$ and variance

$$
\begin{aligned}
\sigma^2_{\mathbf{x}}(a, \hat{a}^*_{\mathbf{x}}) &= \frac{\sigma^2(\mathbf{x}, \hat{a}^*_{\mathbf{x}})}{N(\mathbf{x}, \hat{a}^*_{\mathbf{x}})} + \frac{\sigma^2(\mathbf{x}, a)}{N(\mathbf{x}, a)} \\
&= \frac{1}{N(\mathbf{x}, \hat{a}^*_{\mathbf{x}})} \left(\sigma^2(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) + \sigma(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) \sigma^2(\mathbf{x}, a) \sqrt{\sum_{\tilde{a} \in A_{\mathbf{x}}, \tilde{a} \neq \hat{a}^*_{\mathbf{x}}} \frac{(r_{\mathbf{x}}(\tilde{a}, a))^2}{\sigma^2(\mathbf{x}, \tilde{a})}} \right).
\end{aligned}
$$

Apply inequality $\sqrt{\sum_{i=1}^n c_i^2} \leq \sum_{i=1}^n \sqrt{c_i^2} = \sum_{i=1}^n c_i$ for positive numbers c_i's yields

$$
\sigma^2_{\mathbf{x}}(a, \hat{a}^*_{\mathbf{x}}) \leq \frac{1}{N(\mathbf{x}, \hat{a}^*_{\mathbf{x}})} \left(\sigma^2(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) + \sigma(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) \sigma^2(\mathbf{x}, a) \sum_{\tilde{a} \in A_{\mathbf{x}}, \tilde{a} \neq \hat{a}^*_{\mathbf{x}}} \frac{r_{\mathbf{x}}(\tilde{a}, a)}{\sigma(\mathbf{x}, \tilde{a})} \right).
$$

Since APCS is decreasing in $\sigma^2_{\mathbf{x}}(a, \hat{a}^*_{\mathbf{x}})$, we have

$$
PCS \geq 1 - \sum_{a \in A_{\mathbf{x}}, a \neq \hat{a}^*_{\mathbf{x}}} \Phi\left(- \frac{\delta_{\mathbf{x}}(\hat{a}^*_{\mathbf{x}}, a) \sqrt{N(\mathbf{x}, \hat{a}^*_{\mathbf{x}})}}{\sqrt{\sigma^2(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) + \sigma(\mathbf{x}, \hat{a}^*_{\mathbf{x}}) \sigma^2(\mathbf{x}, a) \sum_{\tilde{a} \in A_{\mathbf{x}}, \tilde{a} \neq \hat{a}^*_{\mathbf{x}}} \frac{r_{\mathbf{x}}(\tilde{a}, a)}{\sigma(\mathbf{x}, \tilde{a})}}} \right)
$$

as desired. □

3.5 Numerical Examples

In this section, we evaluate our proposed OCBA-MCTS on two tree search problems against the well-known UCT [45]. The effectiveness is measured by PCS, which is estimated by the fraction of times the algorithm chooses the true optimal action. We first evaluate our algorithm on an inventory control problem with random non-normal reward. Then we apply our algorithm to the game of tic-tac-toe.

For convenience, we restate the UCT tree policy here. At a state node $\mathbf{x}$, the UCT policy will select the child state-action node with the highest upper confidence bound, i.e.,

$$\hat{a} = \arg\max_{a \in A_x} \left\{ \bar{Q}(\mathbf{x}, a) + w_e \sqrt{\frac{2 \log \sum_{a' \in A_x} N(\mathbf{x}, a')}{N(\mathbf{x}, a)}} \right\}, \tag{3.26}$$

where w_e is the "exploration weight". The original UCT algorithm assumes the value function in each stage is bounded in $[0,1]$ because it sets $w_e = 1$, whereas the support is unknown in many practical problems. Therefore, in general, w_e needs to be tuned to encourage exploration.

For all experiments, we set the smoothing parameter in Equation (3.13) in the back-propagation phase to $\alpha_{N(\mathbf{x})} = 1 - \frac{1}{5N(\mathbf{x})}$. Since initial estimates of sample variance can be less accurate with small n_0, we add an initial variance $\sigma_0^2 > 0$, which decays as the number of visits grows, to the sample variance to encourage exploration. Specifically, we

set

$$\hat{\sigma}^2(\mathbf{x},a) = \frac{1}{N(\mathbf{x},a)} \sum_{t=1}^{N(\mathbf{x},a)} \left(\hat{Q}^t(\mathbf{x},a) - \bar{Q}(\mathbf{x},a)\right)^2 + \sigma_0^2/N(\mathbf{x},a),$$

where the first term is the sample variance, and second term vanishes as $N(\mathbf{x},a)$ grows.

3.5.1 Inventory Control Problem

We now evaluate the performance of OCBA-MCTS using the inventory control problem in [43]. The objective is to find the initial order quantity that minimizes the total cost over a finite horizon. At decision period i, we denote by D_i the random demand in period i, $\mathbf{x}_i = (x_i, i)$ the state node, where x_i is the inventory level at the end of period i (which is also the inventory at the beginning of period $i+1$), $(\mathbf{x}_i, a_i)$ the corresponding child state-action node with a_i being the order amount in period i, p the per period per unit demand lost penalty cost, h the per period per unit inventory holding cost, K the fixed (set-up) cost per order, M the maximum inventory level (storage capacity) and H the number of simulation stages. We set $M = 20$, initial state $x_0 = 5$, $h = 1$, $H = 3$, $D_i \sim DU(0,9)$ (discrete uniform, inclusive), and consider two different settings for p and K:

1. Experiment 1: $p = 10$ and $K = 0$;

2. Experiment 2: $p = 1$ and $K = 5$.

The reward function, which in this case is the negative of the inventory cost in stage i, is defined by

$$R(\mathbf{x}_i, a_i) = -\left(h\max\{0, x_i + a_i - D_i\} + p\max\{0, D_i - x_i - a_i\} + K\mathbb{1}_{\{a_i>0\}}\right),$$

where $\mathbb{1}$ is the indicator function, and the state transition follows

$$x_{i+1} = \max(0, x_i + a_i - D_i),$$

where

$$a_i \in A_{x_i} = \{a | x_i + a \leq M\}.$$

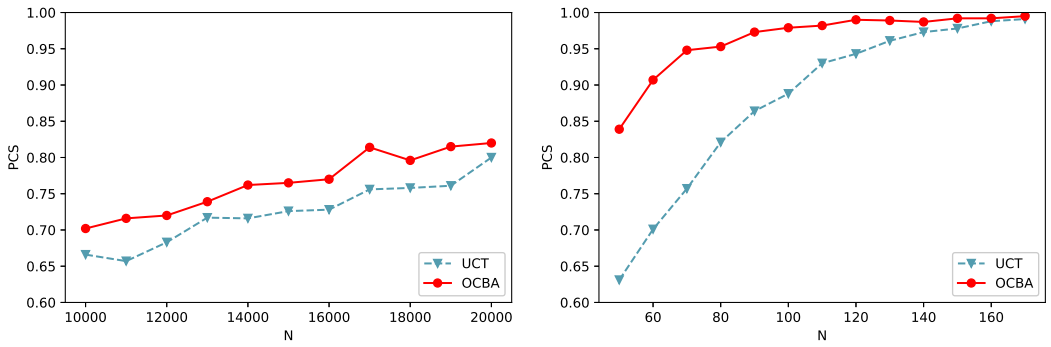
For UCT, to accommodate the reward support not being $[0,1]$, we adjust the exploration weight when updating a state-action node, i.e., set w_e initially to 1, then in the backpropagation step, update w_e by

$$w_e = \max(w_e, |\hat{Q}^{N(\mathbf{x},a)}(\mathbf{x}, a)|),$$

where $\hat{Q}^{N(\mathbf{x},a)}(\mathbf{x}, a)$ is obtained in Equation (3.10). The initial variance σ_0^2 is set to 100. For both OCBA-MCTS and UCT, we set the number of expansions (n_0) to 4 for depth 1 state-action nodes (i.e., the child nodes of the root) and to 2 for all other state action nodes in Experiment 1, and set n_0 to 2 for all nodes in Experiment 2. The different values of n_0 are due to the variance decreasing with the depth of a node, and Experiment 2 is a

relatively easier problem.

For both experiment settings, each algorithm is repeated $1,000$ times at each simulation budget level N to estimate PCS. Since Experiment 1 is a much harder problem compared to Experiment 2, more rollouts (budget) are required. Therefore, N ranges from $10,000$ to $20,000$ and from 50 to 170 for Experiments 1 and 2, respectively.



(a) Experiment 1: $p = 10$, $K = 0$ (b) Experiment 2: $p = 1$, $K = 5$

Figure 3.1: The estimated PCS as a function of sampling budget achieved by UCT-MCTS and OCBA-MCTS for inventory control problem, averaged over 1,000 runs.

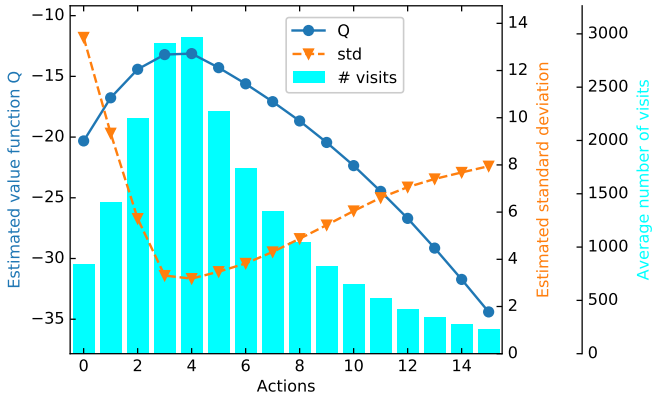
The estimated PCS curves for both experiments are illustrated in Figure 3.1, where the standard error ($= \sqrt{PCS(1-PCS)/N}$) is small and thus omitted for clarity. OCBA-MCTS achieves better PCS for both experiment setups. For Experiment 1 (optimal action $a_0^* = 4$), as shown in Figure 3.1a, OCBA-MCTS achieves a 10% higher PCS (absolute) compared to UCT. For Experiment 2 (optimal action $a_0^* = 0$), we see a 20% performance gap between UCT and OCBA-MCTS when the number of samples is less than 100, after which UCT gradually closes the gap as expected.

It is also beneficial to compare the distribution of budget allocation of OCBA-MCTS and UCT to show the exploration-exploitation balance of OCBA-MCTS. For convenience, we label the child actions of the root node from 0 to 15, where action i denotes

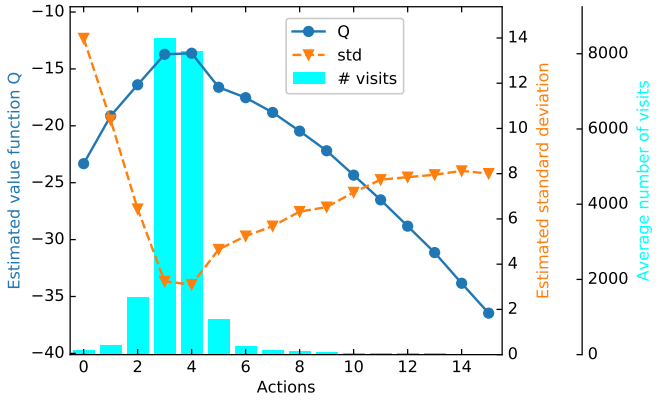
ordering i units. Figures 3.2 and 3.3 illustrate the average number of visits, average estimated value function, and average estimated standard deviation of all child state-action nodes of the root node over 1,000 repeated runs with 20,000 and 170 rollouts for Experiment 1 and Experiment 2, respectively. Note that although the estimated standard deviation does not play a role in determining the allocation for UCT, we still plot it for reference. Both figures show that the number of visits to children nodes is, to some extent, proportional to the estimated value of the node for UCT. On the other hand, OCBA-MCTS puts more effort on the estimated optimal and second optimal actions (actions 4 and 3 for Experiment 1 and actions 0 and 1 for Experiment 2, respectively), as illustrated in Figures 3.2b and 3.3b.

In Experiment 1 where there are two competing actions with similar estimated values (actions 3 and 4, with action 4 being the optimal), OCBA-MCTS will spend most of its sampling budget on those two potential actions and put much lesser effort on clearly inferior actions, such as actions 6 to 14, compared to UCT. This strategy makes more sense when the objective is to identify the best action, and thus is more suitable for MCTS problems, as the ultimate goal is to make a decision. It is also interesting to note that OCBA-MCTS actually allocates slightly more visits to the competing suboptimal action than the optimal one (mean 8486 and 8468 for actions 3 and 4, respectively), which will not happen in bandit-based policies, as their goal is to minimize regret, and thus will put more effort on exploiting the estimated optimal action. In Experiment 2 where the optimum is slightly easier to find, although OCBA-MCTS allocates a larger fraction of samples to suboptimal actions compared to that in Experiment 1, most of the samples are still allocated to the top 2 actions as shown in Figure 3.3b, whereas UCT performs similar

to that in Experiment 1.



(a) UCT

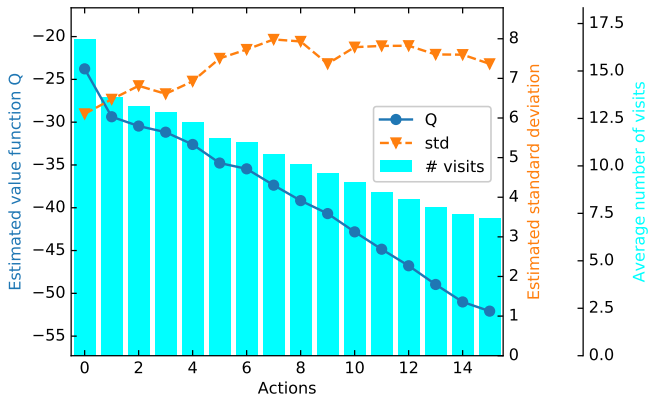


(b) OCBA

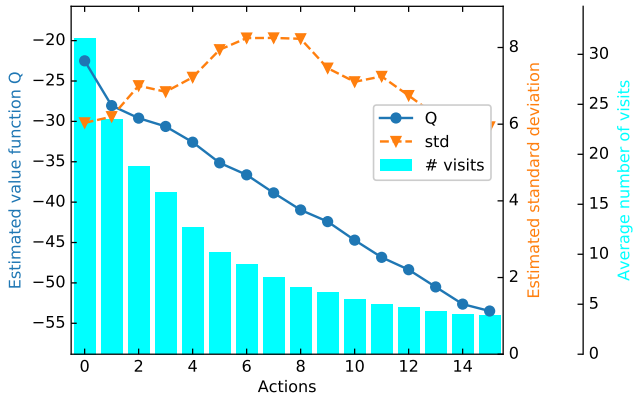
Figure 3.2: Sampling distribution for Experiment 1 with $N = 20,000$, averaged over 1,000 runs.

3.5.2 Tic-Tac-Toe

In this section, we apply OCBA-MCTS and UCT to the game of tic-tac-toe to identify the optimal move. Tic-tac-toe is a game for two players who take turns marking 'X' (Player 1) and 'O' (Player 2) on a 3×3 board. The objective for Player 1 (Player 2) is to mark 3 consecutive 'X' ('O') in a row, column or diagonal. If both players act optimally, the game will always end in a draw.



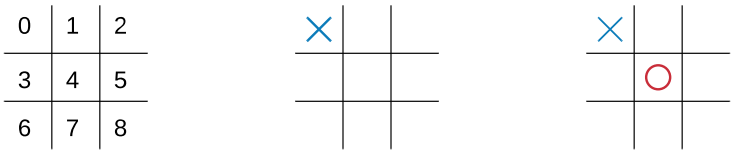
(a) UCT



(b) OCBA

Figure 3.3: Sampling distribution for Experiment 2 with $N = 170$, averaged over 1,000 runs.

For ease of presentation, we number the spaces sequentially as shown in Figure 3.4a. We use OCBA-MCTS and UCT to represent Player 2, with Player 1 marked 'X' on space 0 as shown in Figure 3.4b. In this situation, the optimal move for Player 2 will be marking space 4 (shown in Figure 3.4c), as taking any other space will end up in losing the game if Player 1 plays optimally. In this game, Player 2 (MCTS algorithm) makes decisions at even stages $(0, 2, 4, \dots)$ and Player 1 makes decisions at odd stages $(1, 3, \dots)$. The state transitioning is deterministic and Player 1's move is modeled using a randomized policy. We consider two different policies for Player 1:



(a) Action layout (b) Root node (c) Optimal

Figure 3.4: Tic-tac-toe board setup.

1. Experiment 3: Player 1 plays randomly, i.e., with equal probability to mark any feasible space;

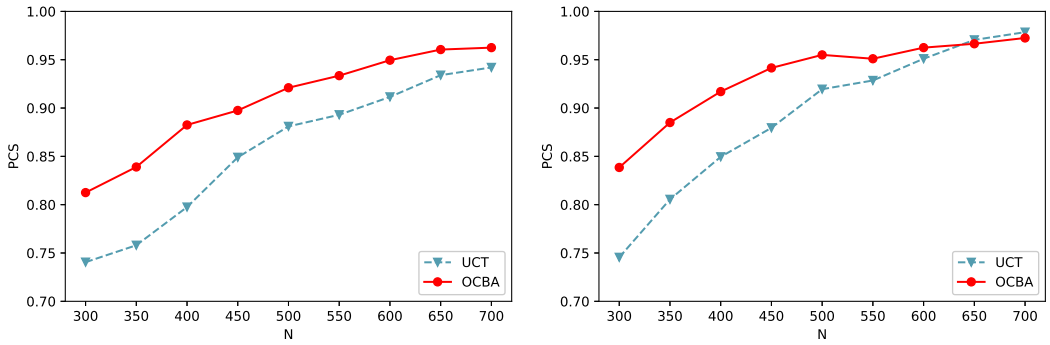
2. Experiment 4: Player 1 plays UCT.

We compare the performance of OCBA-MCTS and UCT on Player 2 in both experiments. At state node $\mathbf{x}$, the reward function for taking action a is defined according to the following rules: immediately after taking the action, if Player 2 wins the game, $R(\mathbf{x},a) = 1$, if it leads to a draw, $R(\mathbf{x},a) = 0.5$; otherwise (Player 2 loses or in any non-terminating state), $R(\mathbf{x},a) = 0$. n_0 is set to 2 across all nodes for both UCT and OCBA-MCTS. Since the value function for all state-action nodes is now bounded in $[0,1]$, we set $w_e = 1$ throughout the entire experiment for UCT policies. The initial variance σ_0^2 is set to 10. For Experiment 4 where Player 1 plays UCT, its goal is to *minimize* the reward, therefore, Player 1 will select the action that minimizes the lower confidence bound, i.e.,

$$\hat{a} = \arg\min_{a \in A_x} \left\{ \bar{Q}(\mathbf{x},a) - w_e \sqrt{\frac{2\log \sum_{a' \in A_x} N(\mathbf{x},a')}{N(\mathbf{x},a)}} \right\}.$$

Similar to the previous section, we plot the PCS of the two algorithms as a function of the number of rollouts, which ranges from 300 to 700 for both experiments and the PCS is estimated over 2000 independent experiments at each rollout level. The results are

shown in Figure 3.5, which indicates that the proposed OCBA-MCTS produces a more accurate estimate of the optimal action compared to UCT. Both experiments show that



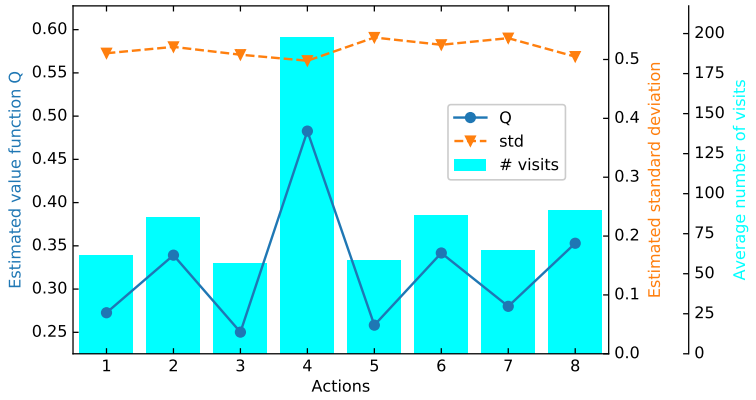
(a) Experiment 3: Player 1 plays randomly. (b) Experiment 4: Player 1 plays UCT.

Figure 3.5: The estimated PCS as a function of sampling budget achieved by UCT-MCTS and OCBA-MCTS for tic-tac-toe, averaged over 2000 runs.

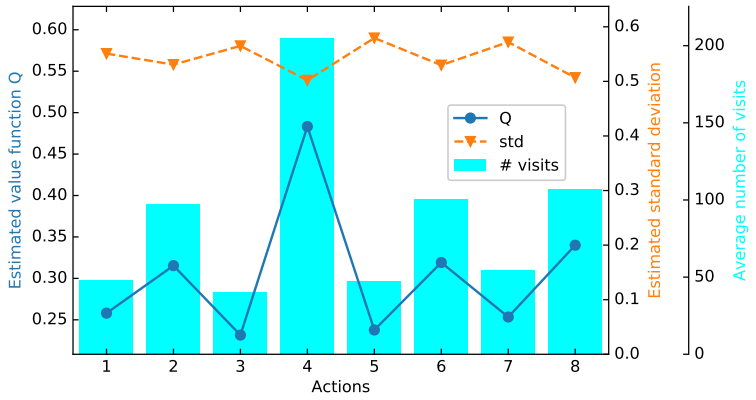
OCBA-MCTS is better at finding the optimal move when the sampling budget is relatively low. The performance of UCT and OCBA-MCTS become comparable when more samples become available. We also note that there is a greater performance gap between UCT and OCBA-MCTS in Experiment 3 than in Experiment 4: in Experiment 3, OCBA-MCTS achieves 10% better PCS, whereas in Experiment 4, the difference is around 5% when $N < 500$ and soon catches up as N increases. This is expected, as it becomes easier to determine the optimal action when the opponent applies an AI algorithm (i.e., Player 1 has a better chance to take its optimal action). In this case, space 4 becomes a clear optimum and therefore Player 2's UCT algorithm tends to exploit it more, which leads to better performance.

The sampling distributions for OCBA-MCTS and UCT with $N = 700$ for both experiments are shown in Figures 3.6 and 3.7. In this game, since a relatively clear optimum is available, OCBA-MCTS and UCT behaved differently compared to that in the inventory

control problem. As shown in Figures 3.6a and 3.7a, UCT spends most of the sampling budget exploiting this action, whereas OCBA will still try to explore other suboptimal actions due to its tendency to better balance exploration and exploitation.



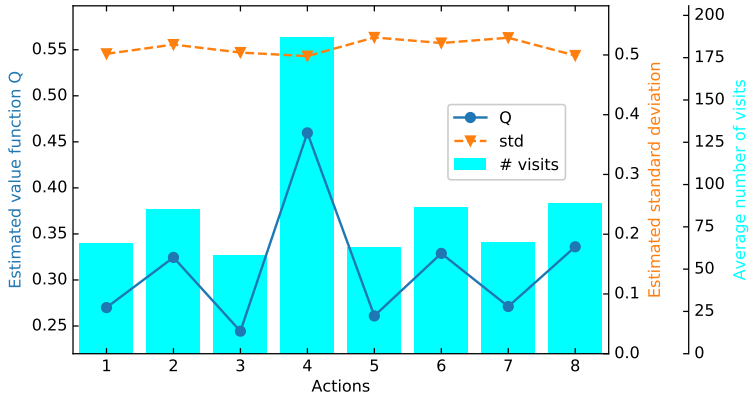
(a) UCT



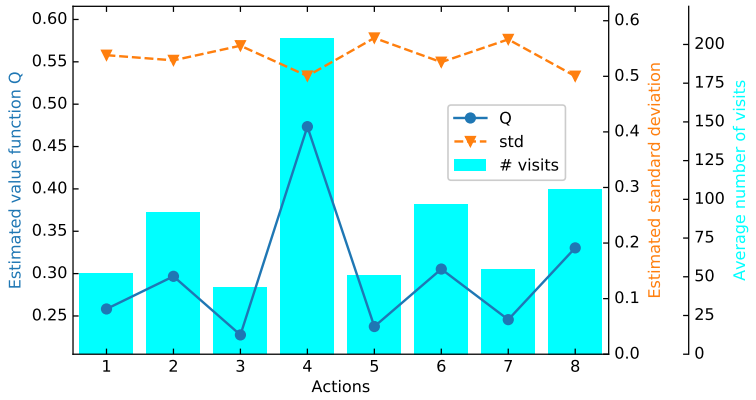
(b) OCBA

Figure 3.6: Sampling distributions for Experiment 3, averaged over 2000 runs.

In summary, the proposed OCBA-MCTS outperforms UCT in both experiments in finding the optimal action at the root. Since the objective of the proposed OCBA tree policy is to maximize PCS, it leads to different budget allocation and better PCS.



(a) UCT



(b) OCBA

Figure 3.7: Sampling distributions for Experiment 4, averaged over 2000 runs.

3.6 Conclusion and Future Research

In this research, we present a new OCBA tree policy for MCTS. Unlike bandit-based tree policies (e.g., UCT), the new policy maximizes PCS at the root node, and in doing so, balances the exploration and exploitation trade-off differently. Furthermore, the new OCBA tree policy relaxes the assumption of known bounded support on the reward distribution, and thus makes MCTS more generally applicable.

For future research, we intend to explore the use of a batch sampling scheme in Algorithm 4, which allocates a batch of $\Delta > 1$ samples at each node. With batch sampling

and updating, we may exploit the power of parallel computing to more quickly identify the optimal action. Furthermore, the proposed OCBA-MCTS algorithm aims at selecting only the optimal action at the root. It may be of interest to show that our algorithm is $\varepsilon - \delta$-correct and establish an upper bound on the sample complexity. It is also interesting to consider cases where the optimal and the suboptimal actions have similar Q-values (such as that in Experiment 1) and the decision maker is insensitive to selecting the slightly suboptimal actions. Under such settings, the "indifference zone" approaches could be applied.

Chapter 4: Parameter Estimation

4.1 Introduction

The human brain continuously processes complex information it receives, therefore, decoding the dynamics of brain activities underlying conscious behavior is one of the key questions in systems neuroscience. To quantify the brain activities, neurolimaging modalities, such as electroencephalography (EEG) and magnetoencephalography (MEG), are widely used. The general framework to model the neural dynamics with auditory stimuli is shown in Figure 4.1. Since the M/EEG measurements usually have millisecond temporal resolution, the modeling of brain activities must be of comparable temporal resolution to that of M/EEG acquisition.

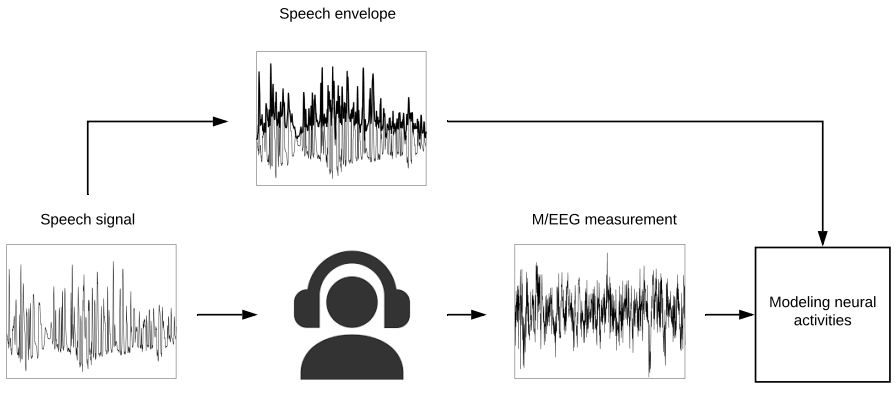


Figure 4.1: General framework to model neural dynamics.

In the neuroscience literature, many static estimation methods are proposed to characterize the neural response functions. For example, "reverse correlation" [69–71] and boosting [72–74] are two widely used techniques to construct neural models, where the neural responses are averaged over a long time (typically at the scale of a minute [75]) and trials of experiment to form reliable response functions. However, it is shown that sensory neurons, such as those in the auditory system, can undergo rapid changes in their response characteristics, and thereby result in functional changes over time [76–80]. In addition, [81] reveals that task-based behavioral and neural plasticity in the auditory cortex can occur in less than a second. Therefore, static methods are not suitable to systematically track such neural plasticity at the order of a second, and thus, a dynamic framework becomes crucial to better understand the underlying neural cognitive functions.

To address this issue, linear system theory is utilized to model the neural activities using the measured M/EEG data with proven record [82–84]. Under this setting, a linear dynamics of the neural activities is assumed and is used to predict the M/EEG response by convolving with the features of the stimuli (such as speech). The neural activities, which in this case is modeled as impulse response functions with linear state space model, are crucial in characterizing the temporal structure of auditory information processing in the brain, and are often referred to as Temporal Response Functions (TRF) [85–87].

The models (both latent state process and the M/EEG prediction process) are often corrupted with noise, whose parameters are estimated from the M/EEG observations. In many applications, the noise terms are assumed to be independent and identically distributed (i.i.d.) Gaussian, which would result in convenient closed-form solutions [83, 88, 89]. However, neural response models such as TRF are observed to contain

two major peaks near 50 ms (M_{50}) and 100 ms (M_{100}) that are modulated by different features of the speech stimuli [75]. As a result, simple i.i.d. Gaussian state input noise would seem insufficient to model such characteristics. In addition, the independent assumption of the noise term would largely lose track of the temporal correlations in the component.

To address these issues, one solution is to assume underlying low dimensional noise processes, which are then linearly mapped into TRF space with a set of base vectors. The TRF characteristics, such as the aforementioned M_{50} and M_{100}, are represented by the base vectors. The underlying noise process can be used to model the temporal correlation, hence autoregressive (AR) processes driven by i.i.d. Gaussian become a natural choice. The model parameters can then be estimated by maximum likelihood. On the other hand, although such a modeling approach would better characterize the neural activities, it comes at the expense of a more complicated likelihood function to be optimized, which is usually non-convex without a closed form. The complexity in the likelihood function can be addressed by Expectation-Maximization (EM) algorithm [88, 90]; however, a global optimization method is needed to resolve this issue, as the cost function in the Maximization step can still be nonconvex without a closed-form update.

One of the most commonly employed global optimization approaches, especially when there is a general lack of model information, is random search, which is further categorized as instance-based or model-based [91]. Instance-based methods generate new sample points based on current points, such as simulated annealing and genetic algorithms. In this work, we will focus on applying model-based algorithms, where candidate points rely on an underlying distribution whose parameter is updated every iteration. Examples of model-based search algorithm include the cross-entropy (CE) method [92, 93]

and ant colony optimization (ACO) [91, 94].

Model-based random search algorithms facilitate solving the problem of interest in various ways. First, since probability models are used to guide the construction of candidate solutions in model-based methods, they are easy to implement. Second, model-based methods can be customized to solve a particular type of problem with convergence assurance. In addition, current methods for model parameter estimations only provide point parameter estimates, where the features of the parameters are lost, whereas model-based methods maintain the distribution of the parameter estimates.

In summary, we propose a state space model to represent the dynamics of the neural activities, where the state noise terms are modeled by AR processes and subsequently mapped to high dimensions by base matrix to characterize neural activities. The EM algorithm is employed to estimated the unknown parameters, where the Maximization step is carried out with a model-based global optimization method to deal with nonconvexity of the objective function. Finally, experiments with both synthesized data and acquired MEG data are carried to demonstrate the efficiency of the proposed algorithm.

4.2 Problem Formulation and Proposed Solution

4.2.1 Problem Formulation

Consider the following state space model

$$x_t = Fx_{t-1} + w_t, \tag{4.1}$$

$$y_t = C_t x_t + v_t, \tag{4.2}$$

where $x_t \in R^m$ represents the neural activities, $w_t \in R^m$ is a non-Gaussian state input noise with temporal correlation, $y_t \in R^n$ is the observation, and $v_t \in R^n$ is i.i.d. Gaussian observation noise with $v_t \sim N(0,V)$, all at time step t. For convenience, we define $x_{1:T} = \{x_1, x_2, \ldots, x_T\}$, and $y_{1:T}$ is defined analogously. The time-invariant transfer matrix $F \in R^{m \times m}$, which can be set with domain specific knowledge, the measurement matrix $C_t \in R^{n \times m}$, which represents the neural stimuli and thus can be estimated, and the observation noise covariance V are assumed to be known.

Remark 4.1. *We assume matrices F and V to be known, as they can either be set with prior information or estimated otherwise by measurements for convenience; however, these assumptions are not necessary, as they can also be estimated with the proposed EM algorithm along with other unknown parameters.*

We model state noise w_t by a set of AR(p) processes ($p \geq 1$):

$$z_{t,i} = \sum_{l=1}^{p} \phi_{i,l} z_{t-j,i} + \varepsilon_{t,i}, \tag{4.3}$$

$$w_t = \sum_{i=1}^{h} \alpha_i z_{t,i},\ i = 1, 2, \ldots, h \tag{4.4}$$

where $z_{t,i} \in R$ is the i-th AR(p) process with unknown parameters $\phi_{i,l}$, $\varepsilon_{t,i} \sim N(0, \sigma_i^2)$ with unknown variances σ_i^2, and $\alpha_i \in R^m$ is a set of unknown bases.

In practical settings, the dimension of the state (i.e., m) may be large, hence it can be challenging to estimate all components of the bases. A more practical solution would be instead assuming an underlying dictionary (e.g., Gaussian dictionary) that constitutes

the bases, i.e., let

$$D = [d_1, d_2, \ldots, d_k],\ d_i \in R^m,\ \forall i,$$

be the set of known dictionaries with $k \ll m$. The bases are constructed by

$$\alpha_i = \sum_{j=1}^{k} d_j n_{ji},$$

where the weights n_{ji} for the i-th base and j-th dictionary are unknown.

Under these settings, the AR processes model parameters $\phi_{i,l}$, AR process white noise variances σ_i^2, and weights n_{ji} are unknown (totaling $h(p+k+1)$ parameters). For convenience, let θ be the vector of unknown parameters. Our objective is to estimate θ from observations $y_{1:T}$ by maximizing its log-likelihood $L(y_{1:T};\theta)$, which, however, is hard to compute. Therefore, one possible solution is to use EM. Formally, define the log-likelihood of $y_{1:T}$ and joint state-observation log-likelihood of $x_{1:T}$ and $y_{1:T}$ with estimated underlying parameters θ by $L(y_{1:T};\theta)$ and $L(x_{1:T}, y_{1:T};\theta)$, respectively. In the $(m+1)$-th EM iteration, the expected joint state-observation log-likelihood, given an estimate of parameters θ^m and the observations $y_{1:T}$ is calculated (E-step):

$$Q(\theta|\theta^m) = \mathbb{E}[L(x_{1:T}, y_{1:T};\theta)|y_{1:T}, \theta^m], \tag{4.5}$$

where the expectation is taken with respect to (w.r.t.) $x_{1:T}$, given observations $y_{1:T}$ and current parameter estimate θ^m. Then, Equation (4.5) is maximized (M-step) to obtain the

new parameter estimate for the next iterations:

$$\theta^{m+1} = \arg\max_{\theta} Q(\theta|\theta^m), \tag{4.6}$$

it can be shown [90] that the log-likelihood is non-decreasing, i.e.,

$$L(y_{1:T};\theta^{m+1}) \geq L(y_{1:T};\theta^m).$$

When the noise terms are i.i.d. Gaussian, the joint likelihood $L(x_{1:T}, y_{1:T};\theta)$ can be calculated by the Markovian property of the state space model, which is no longer true when the noise is driven by AR processes:

$$L(y_{1:T}, x_{1:T};\theta) = L(y_{1:T}|x_{1:T};\theta)L(x_{1:T};\theta), \tag{4.7}$$

where the first term is straightforward from Equation (4.2); however,

$$L(x_{1:T}) = L(x_0;\theta)\prod_{t=1}^{T} L(x_t|x_{1:t-1};\theta) \tag{4.8}$$

$$\neq L(x_0;\theta)\prod_{t=1}^{T} L(x_t|x_{t-1};\theta), \tag{4.9}$$

as w_t in Equation (4.1) is not i.i.d. Since $\prod_{t=1}^{T} f(x_t|x_{1:t-1};\theta)$ is hard to compute directly, one possibility is to simplify it to $\prod_{t=1}^{T} L(x_t|x_{t-1};\theta)$ to make the problem tractable.

4.2.2 Proposed Solution

We propose to augment the state space model to make the noise term i.i.d. Gaussian. If we denote $z_t = [z_{t,1}, \ldots, z_{t,h}]^T \in R^h$ and $\varepsilon_t = [\varepsilon_{t,1}, \ldots, \varepsilon_{t,h}]^T \in R^h$, then Equation (4.3) can be written compactly as

$$z_t = \sum_{j=1}^{p} \Phi_j z_{t-j} + \varepsilon_t$$

$$\varepsilon_t \sim N(0, \Sigma),$$

where

$$\Phi_j = diag(\phi_{1,j}, \ldots, \phi_{h,j}) \in R^{h \times h}, \forall\ j = 1, \ldots, p$$

$$\Sigma = diag(\sigma_1^2, \ldots, \sigma_h^2) \in R^{h \times h}.$$

Also denote $A = [\alpha_1, \ldots, \alpha_h] \in R^{m \times h}$ (henceforth referred to as base matrix), Equation (4.4) can be rewritten as

$$\begin{aligned} w_t &= A z_t \\ &= \sum_{j=1}^{p} A \Phi_j z_{t-j} + A \varepsilon_t. \end{aligned}$$

And if we define $N = [n_{ji}]_{m\times k} = [n_1, \ldots, n_k]$, where n_i is the i-th column of N, (i.e., the weights for the i-th base), then A can also be expressed compactly as

$$A = DN = D[n_1, \ldots, n_k].$$

The state space model in Equation (4.1) can be reformulated as

$$\begin{aligned}
\tilde{x}_t = \begin{bmatrix} x_t \\ z_t \\ z_{t-1} \\ \vdots \\ z_{t-p+1} \end{bmatrix} &= \begin{bmatrix} F & DN\Phi_1 & \ldots & DN\Phi_{p-1} & DN\Phi_p \\ \mathbf{0} & \Phi_1 & \ldots & \Phi_{p-1} & \Phi_p \\ \mathbf{0} & I & \ldots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ldots & I & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_{t-1} \\ z_{t-1} \\ z_{t-2} \\ \vdots \\ z_{t-p} \end{bmatrix} + \begin{bmatrix} DN \\ I \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \varepsilon_t \\
&= \tilde{F}\tilde{x}_{t-1} + \Gamma\varepsilon_t \\
&= \tilde{F}\tilde{x}_{t-1} + \tilde{w}_t,
\end{aligned} \tag{4.10}$$

where $\mathbf{0}$ and I, respectively, denote the zero matrix and identity matrix of appropriate size. The noise term

$$\tilde{w}_t \sim N(0, \tilde{\Sigma})$$

$$\tilde{\Sigma} = \Gamma^T \Sigma \Gamma$$

is i.i.d. It is worth noting that, in general, the new covariance matrix $\tilde{\Sigma}$ is not of full rank, as there are fewer number of AR processes (which determines the rank) than the state di-

mension (which is the dimension of $\tilde{\Sigma}$). Therefore, the state space model in Equation (4.1) and 4.2 can be written as

$$\tilde{x}_t = \tilde{F}\tilde{x}_{t-1} + \tilde{w}_t \tag{4.11}$$

$$y_t = H_t\tilde{x}_t + v_t, \tag{4.12}$$

with

$$H_t = [C_t, 0, \ldots, 0].$$

4.3 Maximum Likelihood Estimation of the Unknown Parameters

In this section, we compute the maximum likelihood estimator (MLE) of θ from observations $y_{1:T}$ using EM.

4.3.1 E-Step

Similar to that in Equation (4.7) and 4.8,

$$L(y_{1:T}, \tilde{x}_{1:T}; \theta) = L(y_{1:T}|\tilde{x}_{1:T}; \theta)L(\tilde{x}_{1:T}; \theta).$$

The first term can be can be computed by:

$$\begin{aligned} L(y_{1:T}|\tilde{x}_{1:T};\theta) &= \prod_{t=1}^{T} L(y_t|\tilde{x}_{1:T}, y_{1:t-1};\theta) \\ &= \prod_{t=1}^{T} L(y_t|\tilde{x}_t;\theta), \end{aligned}$$

which is given by the probability desity function (p.d.f.) of $\mathcal{N}(H_t\tilde{x}_t, V)$.

Similarly, if we denote the initial state at time 0 by $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$, the second term can now be simplified since the state noise is i.i.d.:

$$\begin{aligned} L(\tilde{x}_{1:T};\theta) &= L(\tilde{x}_0;\theta)\prod_{t=1}^{T} L(\tilde{x}_t|\tilde{x}_{1:t-1}) \\ &= L(\tilde{x}_0;\theta)\prod_{t=1}^{T} L(\tilde{x}_t|\tilde{x}_{t-1}), \end{aligned}$$

which is given by the product of the p.d.f. of $\mathcal{N}(\tilde{F}\tilde{x}_{t-1}, \tilde{\Sigma})$ for $t = 1, 2\ldots, T$.

Combining the two terms yields

$$L(y_{1:T}, \tilde{x}_{1:T};\theta) = L(\tilde{x}_0;\theta)\prod_{t=1}^{T} f(y_t|\tilde{x}_t;\theta)L(\tilde{x}_t|\tilde{x}_{t-1}).$$

Taking logarithm on both sides yields

$$\begin{aligned} L(\tilde{x}_{1:T}, y_{1:T};\theta) =& \log L(\tilde{x}_0;\theta) + \sum_{t=1}^{T}[\log L(y_t|\tilde{x}_t;\theta) + \log L(\tilde{x}_t|\tilde{x}_{t-1})] \\ =& -\frac{1}{2}\sum_{t=1}^{T}[(y_t - H_t\tilde{x}_t)^T V^{-1}(y_t - H_t\tilde{x}_t) + (\tilde{x}_t - \tilde{F}\tilde{x}_{t-1})^T\tilde{\Sigma}^{\dagger}(\tilde{x}_t - \tilde{F}\tilde{x}_{t-1})] \\ & -\frac{T}{2}\sum_{i=1}^{h}\log\lambda_i + constant, \end{aligned}$$

where $\tilde{\Sigma}^{\dagger}$ is the Moore–Penrose inverse of $\tilde{\Sigma}$ and λ_i is the i-th non-zero eigenvalue of $\tilde{\Sigma}$. If $\tilde{\Sigma}$ is invertible, $\tilde{\Sigma}^{\dagger}$ equals $\tilde{\Sigma}^{-1}$, and $\sum_{i=1}^{h} \log \lambda_i = \det(\tilde{\Sigma})$. Drop the constant term and rewrite the joint log-likelihood:

$$\begin{aligned} -2L(\tilde{x}_{1:T}, y_{1:T}; \theta) = & \sum_{t=1}^{T} [(y_t - H_t \tilde{x}_t)^T V^{-1} (y_t - H_t \tilde{x}_t) + (\tilde{x}_t - \tilde{F} \tilde{x}_{t-1})^T \tilde{\Sigma}^{\dagger} (\tilde{x}_t - \tilde{F} \tilde{x}_{t-1})] \\ & + T \sum_{i=1}^{h} \log \lambda_i. \end{aligned}$$

Let

$$\begin{aligned} L_{t,1} =& (y_t - H_t \tilde{x}_t)^T V^{-1} (y_t - H_t \tilde{x}_t) \\ =& y_t^T V^{-1} y_t - 2 y_t^T V^{-1} H_t \tilde{x}_t + \tilde{x}_t^T H_t^T H_t \tilde{x}_t, \\ L_{t,2} =& (\tilde{x}_t - \tilde{F} \tilde{x}_{t-1})^T \tilde{\Sigma}^{\dagger} (\tilde{x}_t - \tilde{F} \tilde{x}_{t-1}) \\ =& \tilde{x}_t^T \tilde{\Sigma}^{\dagger} \tilde{x}_t - 2 \tilde{x}_{t-1}^T \tilde{F}^T \tilde{\Sigma}^{\dagger} \tilde{x}_t + \tilde{x}_{t-1}^T \tilde{F}^T \tilde{\Sigma}^{\dagger} \tilde{F} \tilde{x}_{t-1}, \end{aligned}$$

and

$$\begin{aligned} \tilde{x}_{t|\tau} =& \mathbb{E}[\tilde{x}_t | y_{1:\tau}; \theta^m], \\ P_{t|\tau} =& \mathbb{E}[(\tilde{x}_t - \tilde{x}_{t|\tau})(\tilde{x}_t - \tilde{x}_{t|T})^T | y_{1:\tau}; \theta^m], \\ P_{t,u|\tau} =& \mathbb{E}[(\tilde{x}_t - \tilde{x}_{t|\tau})(\tilde{x}_u - \tilde{x}_{u|\tau})^T | y_{1:\tau}; \theta^m]. \end{aligned}$$

Then, Equation (4.5) (with constant term dropped) can be written as

$$Q(\theta|\theta^m) = -\frac{1}{2}\sum_{t=1}^{T}\{\mathbb{E}[L_{t,1}|y_{1:T},\theta^m] + \mathbb{E}[L_{t,2}|y_{1:T},\theta^m]\}, \tag{4.13}$$

where

$$\begin{aligned}
\mathbb{E}[L_{t,1}|y_{1:T};\theta^m] =& \mathbb{E}[y_t^T V^{-1} y_t - 2y_t^T V^{-1} H_t \tilde{x}_t + \tilde{x}_t^T H_t^T H_t \tilde{x}_t | y_{1:T};\theta^m] \\
=& y_t^T V^{-1} y_t - 2y_t^T V^{-1} H_t \tilde{x}_{t|T} + \mathbb{E}[tr(\tilde{x}_t^T H_t^T H_t \tilde{x}_t)|y_{1:T};\theta^m] \\
=& y_t^T V^{-1} y_t - 2y_t^T V^{-1} H_t \tilde{x}_{t|T} + \mathbb{E}[tr(\tilde{x}_t \tilde{x}_t^T H_t^T H_t)|y_{1:T};\theta^m] \\
=& y_t^T V^{-1} y_t - 2y_t^T V^{-1} H_t \tilde{x}_{t|T} + tr((P_{t|T} + \tilde{x}_{t|T}\tilde{x}_{t|T}^T) H_t^T H_t),
\end{aligned} \tag{4.14}$$

and

$$\begin{aligned}
\mathbb{E}[L_{t,2}|y_{1:T};\theta^m] =& \mathbb{E}[tr(\tilde{x}_t^T \tilde{\Sigma}^\dagger \tilde{x}_t) - 2tr(\tilde{x}_{t-1}^T \tilde{F}^T \tilde{\Sigma}^\dagger \tilde{x}_t) + tr(\tilde{x}_{t-1}^T \tilde{F}^T \tilde{\Sigma}^\dagger \tilde{F} \tilde{x}_{t-1})|y_{1:T};\theta^m] \\
=& \mathbb{E}[tr(\tilde{x}_t \tilde{x}_t^T \tilde{\Sigma}^\dagger) - 2tr(\tilde{x}_t \tilde{x}_{t-1}^T \tilde{F}^T \tilde{\Sigma}^\dagger) + tr(\tilde{x}_{t-1} \tilde{x}_{t-1}^T \tilde{F}^T \tilde{\Sigma}^\dagger \tilde{F})|y_{1:T};\theta^m] \\
=& tr((P_{t|T} + \tilde{x}_{t|T}\tilde{x}_{t|T}^T)\tilde{\Sigma}^\dagger) - 2tr((P_{t,t-1|T} + \tilde{x}_{t|T}\tilde{x}_{t-1|T}^T)\tilde{F}^T\tilde{\Sigma}^\dagger) \\
& + tr((P_{t-1|T} + \tilde{x}_{t-1|T}\tilde{x}_{t-1|T}^T)\tilde{F}^T\tilde{\Sigma}^\dagger\tilde{F}).
\end{aligned} \tag{4.15}$$

Finally, $\tilde{x}_{t|\tau}$, $P_{t|\tau}$ and $P_{t,u|\tau}$ can be calculated by Kalman filtering and smoothing [88]:

- Kalman filtering (forward)

$$
\begin{aligned}
\tilde{x}_{t+1|t} &= \tilde{F}^m \tilde{x}_{t|t}, \\
\tilde{x}_{t|t} &= \tilde{x}_{t|t-1} + P_{t|t} H_t^T V^{-1} (y_t - H_t x_{t|t-1}), \\
P_{t|t} &= (I - K_t H_t) P_{t|t-1} (I - K_t H_t)^T + K_t V K_t^T, \\
P_{t|t-1} &= \tilde{F}^m P_{t-1|t-1} (\tilde{F}^m)^T + \tilde{\Sigma}^m, \\
K_t &= P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + V)^{-1} \\
&= P_{t|t} H_t^T V^{-1},
\end{aligned}
$$

- Kalman smoothing (backward)

$$
\begin{aligned}
\tilde{x}_{t|T} &= \tilde{x}_{t|t} + J_t (\tilde{x}_{t+1|T} - \tilde{x}_{t+1|t}), \\
J_t &= P_{t|t} (\tilde{F}^m)^T P_{t+1|t}^{-1}, \\
P_{t|T} &= P_{t|t} + J_t (P_{t+1|T} - P_{t+1|t}) J_t^T,
\end{aligned}
$$

- lag-1 covariance smoother

$$
\begin{aligned}
P_{T,T-1|T} &= (I - K_T H_T) \tilde{F}^m P_{T-1|T-1}, \\
P_{t,t-1|T} &= P_{t|t} J_{t-1}^T + J_t (P_{t+1,t|T} - \tilde{F}^m P_{t|t}) J_{t-1}^T,
\end{aligned}
$$

where $\tilde{F}^m$ and $\tilde{\Sigma}^m$ denote the transition matrix and the covariance matrix under parameter θ^m, respectively.

4.3.2 M-Step

Although the E-step can be carried out easily with Equations (4.13) to (4.15), how to maximize it still remains a problem. If the matrices with unknown parameters (e.g., $\tilde{F}$ and $\tilde{\Sigma}$) are decoupled and needed to be estimated in whole (i.e., unconstrained), closed-form update equations are possible [88]; however, the state matrix $\tilde{F}$ and the state input noise covariance $\tilde{\Sigma}$ both depend on the unknown weighting matrix N, and have some special structure as shown in Equation (4.10). As a result, the maximum becomes hard to determine and even nonconvex in general, and a global optimization method is needed to tackle this potentially nonconvex problem.

Model reference adaptive search (MRAS) [95] is an algorithm for solving global optimization problems that works with a parameterized probabilistic model on the solution space and generates at each iteration a group of candidate solutions. In this problem, we apply an underlying Gaussian distribution to model the solution of Equation (4.6). For each MRAS iteration, a set of candidate solutions are generated from the underlying Gaussian distribution. The Q-function value of those candidates are evaluated. Then, a subset of the candidates that have high Q-function values are selected (called the elite set) and are used to update the parameters associated with the Gaussian probabilistic model

by

$$\mu_{k+1} = \frac{\sum_{x \in X^k_{elite}} x[S(Q(x|\theta^m))]^k \exp\left(0.5(x-\mu_k)^T \Sigma_k (x-\mu_k)\right)}{\sum_{x \in X^k_{elite}} [S(Q(x|\theta^m))]^k \exp\left(0.5(x-\mu_k)^T \Sigma_k (x-\mu_k)\right)}, \tag{4.16}$$

$$\Sigma_{k+1} = \frac{\sum_{x \in X^k_{elite}} (x-\mu_{k+1})(x-\mu_{k+1})^T [S(Q(x|\theta^m))]^k \exp\left(0.5(x-\mu_k)^T \Sigma_k (x-\mu_k)\right)}{\sum_{x \in X^k_{elite}} [S(Q(x|\theta^m))]^k \exp\left(0.5(x-\mu_k)^T \Sigma_k (x-\mu_k)\right)}. \tag{4.17}$$

Equations (4.16) to (4.17) will lead the future search biased toward the region containing high-quality solutions, and μ_k converges to the true maximum asymptotically.

After a fixed number of MRAS iterations, a solution to Equation (4.6) is proposed, and the EM algorithm is repeated iteratively until the stopping rule is satisfied. An algorithmic description of the proposed algorithms is given in Algorithms 10 and 11.

Algorithm 10: Expectation-Maximization for parameter estimation.

Input: observations $y_{1:T}$, initial estimate of parameters θ^0, MRAS reference distribution mean μ_0 and variance Σ_0.

Output: estimate of underlying parameters $\hat{\theta}$.

1 Set counter $m \leftarrow 0$.
2 **while** *stopping rule not satisfied* **do**
3 $\quad \theta^{m+1} = MRAS(y_{1:T}, \mu_0, \Sigma_0, \theta^m)$.
4 $\quad m \leftarrow m+1$.
5 Return $\hat{\theta} = \theta^m$

Algorithm 11: MRAS.

Input: observations $y_{1:T}$, MRAS reference distribution with initial mean vector μ_0 and covariance matrix Σ_0, current estimate of parameters θ^m, hyperparameter $\rho_0 \in (0,1]$, initial sample size N_0, $\varepsilon \geq 0$ $\alpha > 0$, mixing coefficient $\lambda \in (0,1]$ and a positive, strictly increasing function $S(\cdot)$.

Output: maximum of $Q(\theta|\theta^m)$.

Set counter $k \leftarrow 0$.

while *stopping rule not satisfied* **do**

  Generate N_k samples $X_1^k, \ldots, X_{N_k}^k$ from $N(\mu_k, \Sigma_k)$ w.p. $(1-\lambda)$ and from $N(\mu_0, \Sigma_0)$ w.p. λ.

  Evaluate the $Q_i^k = Q(X_i^k|\theta^m)$ for $i = 1, \ldots, N_k$ with Equations (4.13) to (4.15).

  Compute the sample $(1-\rho_k)$-quantile:

$$\tilde{\gamma}_{k+1} := Q^k_{(\lceil (1-\rho_k)N_k \rceil)},$$

  where $\lceil \cdot \rceil$ is the ceiling function and $Q^k_{(i)}$ is the i-th ordered statistic of $\{Q_i^k\}$.

  if $k = 0$ *or* $\tilde{\gamma}_{k+1}(\rho_k, N_k) \geq \bar{\gamma}_k + \varepsilon/2$ **then**

    Set $\bar{\gamma}_{k+1} \leftarrow \tilde{\gamma}_{k+1}(\rho_k, N_k)$, $\rho_{k+1} \leftarrow \rho_k$, and $N_{k+1} \leftarrow N_k$.

  else

    Find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{\gamma}_{k+1}(\bar{\rho}, N_k) \geq \bar{\rho}_k + \varepsilon/2$.

    if *such a* $\bar{\rho}$ *exists* **then**

      Set $\bar{\gamma}_{k+1} \leftarrow \tilde{\gamma}_{k+1}(\bar{\rho}, N_k)$, $\rho_{k+1} \leftarrow \bar{\rho}$, and $N_{k+1} \leftarrow N_k$.

    else

      Set $\bar{\gamma}_{k+1} \leftarrow \bar{\gamma}_k$, $\rho_{k+1} \leftarrow \rho_k$, and $N_{k+1} \leftarrow \lceil \alpha N_k \rceil$.

  Find *elite set*

$$\mathscr{X}^k_{elite} = \{X_i^k : Q(X_i^k|\theta^m) \geq \gamma_{k+1}\}.$$

  Update μ_k and Σ_k with Equations (4.16) to (4.17).

  $k \leftarrow k+1$.

Return μ_k.

4.4 Numerical Experiments

In this section, we demonstrate the efficiency of our algorithm by estimating the auditory TRF introduced in Section 4.1. The first experiment is carried out with synthesized data to illustrate the accuracy of estimating the model parameters, and the second is on

real MEG data.

4.4.1 Experiment 1: Synthesized Data

In this experiment, the "observed" data $y_{1:T}$ are generated with the underlying state space model and the AR processes (Equations (4.1) to (4.4)). The observation matrix C_t is given by a measured speech envelope, which is the same speech envelope as that in Experiment 2 to acquire MEG data. We apply the following setup to the experiment with synthesized data.

1. The dimension of TRF is fixed at 24 ($m = 24$).

2. The dictionary size is set to 12 ($k = 12$), with the j-th element of the i-th dictionary given by

$$d_{ij} = f\left(\frac{j - \frac{m}{k}i}{m/(2k)}\right) \; i = 1, \ldots, k, \; j = 1, \ldots, m, \tag{4.18}$$

   where $f(\cdot)$ is the density function of the standard normal distribution. A component of the dictionary is shown in Figure 4.2a, and the i-th column of matrix D is a shift centered at $\frac{m}{k}i$.

3. The order of AR process and the number of AR processes that controls the system noise are both set to 2 ($p, h = 2$). The two AR processes are determined by poles at $p_1 = 0.98\exp(\pm 0.2\pi)$ and $p_2 = 0.95\exp(\pm 0.6\pi)$, i.e., $\phi_{1,1} = 1.58, \phi_{1,2} = -0.587, \phi_{2,1} = 0.960, \phi_{2,2} = -0.902$, and the variances are given by $\sigma_1^2 = 1.2 \times 10^{-8}$ and $\sigma_2^2 = 10^{-8}$ (ground truth).

4. The weights on the dictionary that constitutes the bases are given by (ground truth)

$$N = \begin{bmatrix} -0.0623 & 0.0827 & -0.3450 & -0.7328 & -0.4517 & -0.3594 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3696 & -0.7734 & 0.0519 & -0.5100 & -0.0485 & 0.0098 \end{bmatrix} \tag{4.19}$$

   The two columns of base matrix $A = DN$ (i.e., α_1 and α_2) are shown in Figure 4.3a.

5. The state matrix F in Equation (4.1) is given by $0.998I$, where I is the identity matrix.

6. The number of observed data is 2000 ($T = 2000$).

7. The hyperparameters in MRAS are chosen as: $\alpha = 1.2$, $\rho_0 = 0.1$, $N_0 = 4000$, $\lambda = 0.005$, $\varepsilon = 0$. The MRAS reference distribution is set to be Gaussian with mean θ^m and variance $10^{-4}I$, where I is the identity matrix of same dimension as θ^m. The increasing function $S(\cdot)$ is based on the logistic function:

$$S(x) = \frac{1}{1 + \exp(-\frac{x - \bar{x}}{\max(x) - \min(x)})},$$

   where $\bar{x}$ is the mean of x.

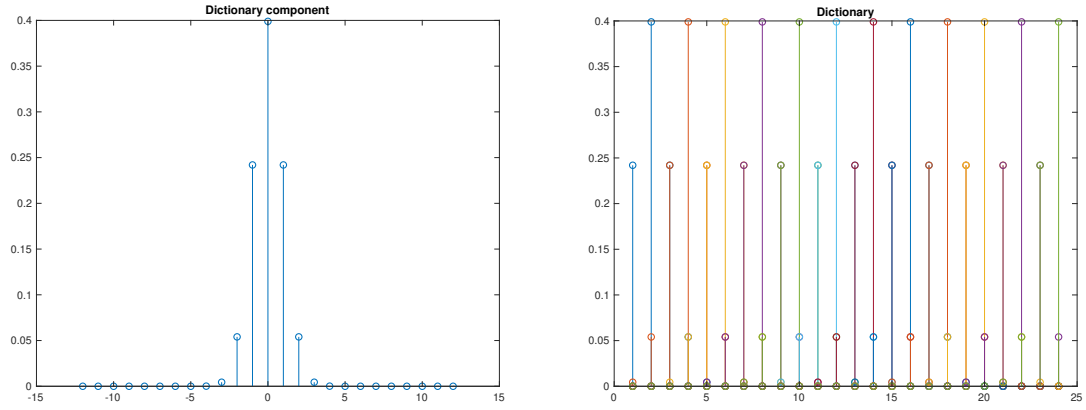
8. The number of MRAS and EM iterations are set to 6 and 8000, respectively.

9. The initial estimate of the parameters (θ^0) is set randomly.

Remark 4.2. *The choices of the model parameters depend on the real MEG data. The MEG data was sampled at* $50Hz$*. Therefore, a 24-dimensional TRF would be sufficient to cover the dynamics of the peaks around* 50 *ms and* 100 *ms. The dictionary, as shown in*

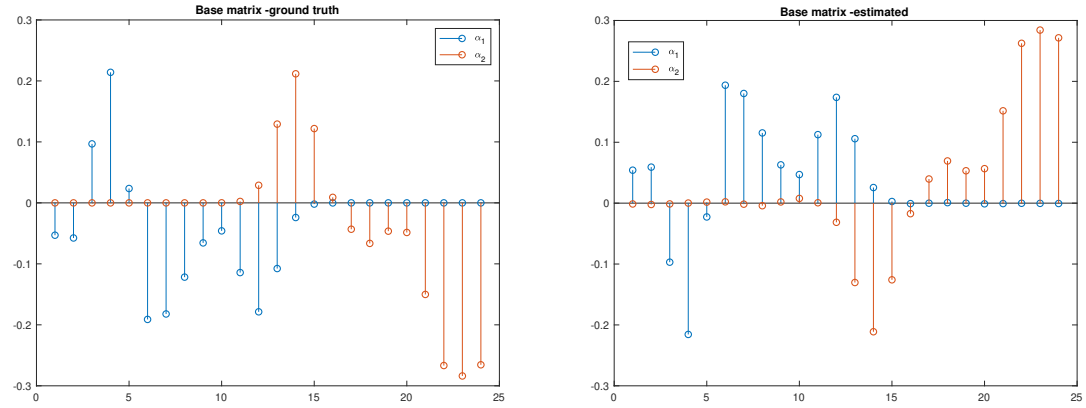
Figure 4.2b, is intentionally set to be sparse to avoid overfitting. The dictionary weights , as shown in Equation (4.19), are set to be non-overlapping to simulate the different parts of TRF being controlled by two AR processes.

Under such settings, there are 30 parameters to be estimated in total.



(a) Component of the dictionary. (b) Dictionary.

Figure 4.2: Dictionary setup.



(a) Ground truth of base matrix. (b) Estimated base matrix

Figure 4.3: Base matrix.

Since our experiments show that the original estimates of the variances of the white noise in AR processes have high variance, we suggest to smooth it with a slid-

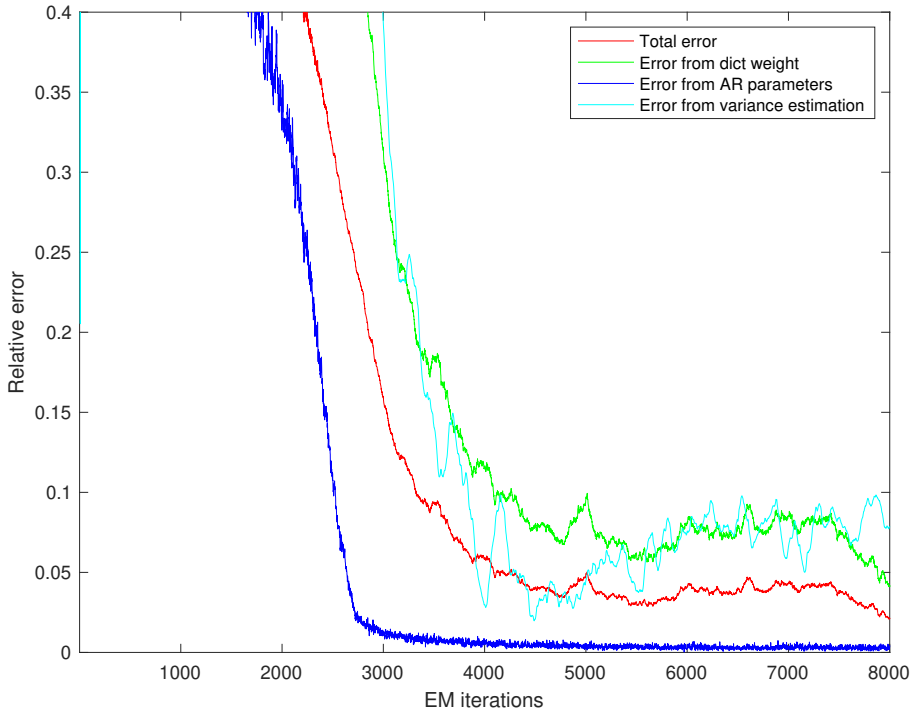


Figure 4.4: Error in parameters estimation.

ing window average of size W, i.e., if we denote the k-th estimate of variance from the EM algorithm by $\sigma_i^{2,k}$, $i = 1,2$, the final smoothed variance estimate is given by $\bar{\sigma}_i^{2,k} = \frac{1}{W}\sum_{j=k-W+1}^{k} \sigma_i^{2,j}$. In this experiment, the window size is set to $W = 100$.

The relative 2-norm errors (i.e., $\frac{\|\hat{\theta}-\theta^*\|_2}{\|\theta^*\|_2}$, where $\hat{\theta}$ and θ^* denote the estimated parameters and ground truth, respectively) of each group of parameters are shown in Figure 4.4. After 5000 EM iterations, each group of parameters converges to the true value within 5% error. For the pole parameters of the AR processes, it takes only 3000 EM iterations to converge.

Figure 4.3b shows the two columns of base matrix from parameter estimates. The proposed algorithm successfully captured the the shape of the bases for their respective AR process despite some overlaps around the 12-th to 15-th dimensions. Note that al-

though the signs of the estimates are flipped compared to the ground truth, we still consider it a correct estimate, as it would be multiplied with the underlying noise process. As a result, it suffices to estimate the relative value and shape within each base.

Figure 4.5 shows the convergence of the weight matrix $N = [n_1, n_2]$, where each curve denotes the error of each weight parameter, i.e., $n_{ij}^k - n_{ij}^*$, where n^k and n^* denote the k-th estimate and the ground truth, respectively. Similar to that in Figure 4.4, the curves converge after around 5000 EM iterations.

4.4.2 Experiment 2: Real MEG data

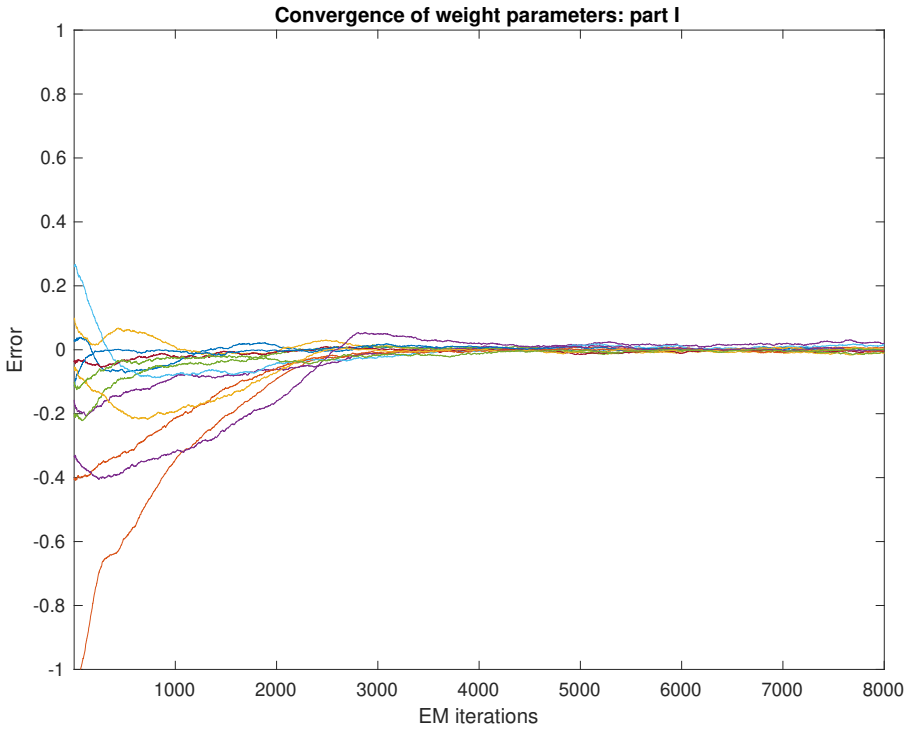
In this experiment, we use the acquired MEG data where a subject listens to a speech. All experiment settings, except the ground truth parameters (which are unknown and to be estimated) and the number of EM iterations (which is set to 50000), are set the same as that in Experiment 1.

Figure 4.6a shows the heatmap of the smoothed states with the estimated parameters, where the y-axis and x-axis denote the each dimension of the states and time steps, respectively. An example of the smoothed states taken at $t = 1500$ is shown in Figure 4.6b. As expected, the smoothed states contain two steady peaks at 50 ms and 100 ms, respectively. The reconstructed base matrix with estimated parameters is displayed in Figure 4.7, which shows that the two AR processes control the dynamics of two regions in a TRF.

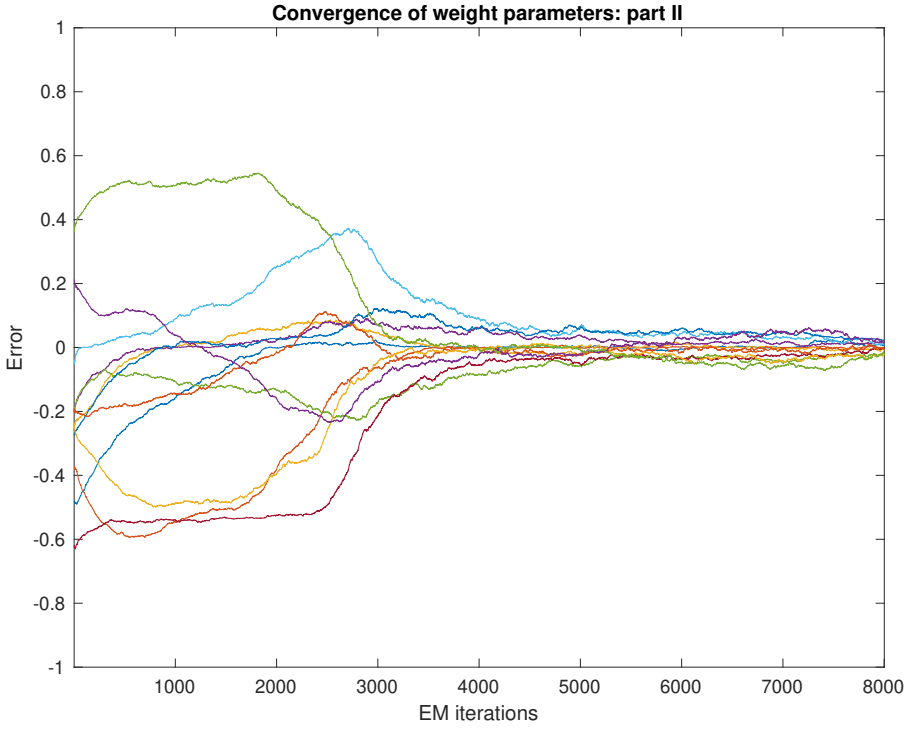
4.5 Conclusion and Future Research

In this work, we propose a linear state space model with non-i.i.d., non-Gaussian, noise process to model the rapid-changing neural dynamics with temporal correlation. To estimate the unknown parameters in the model, ML estimation is carried out with the EM algorithm, where the E-step computed by augmenting the state space model, and a global optimization algorithm is applied to address the general non-convex problem in the M-step. Finally, numerical experiments are carried out to demonstrate the efficiency of the proposed algorithm.

In this work, the non-Gaussian noise model was reformulated as a standard Gaussian model. For future research, we intend to investigate more general noise process, and more adaptive methods such as Gaussian particle filter/smoother can be applied for state estimation. Second, in a real-world scenario, it is more common that a listener attends to a particular speaker and dynamically switches attention in a multi-speaker environment. Thus, it would be meaningful to extend our results to address such "cocktail party" problems. In addition, we model the unknown parameters in this problem as fixed unknown constants, and an ML estimation algorithm is proposed. Another possibility is to treat them as r.v.'s and develop maximum a postori estimators.

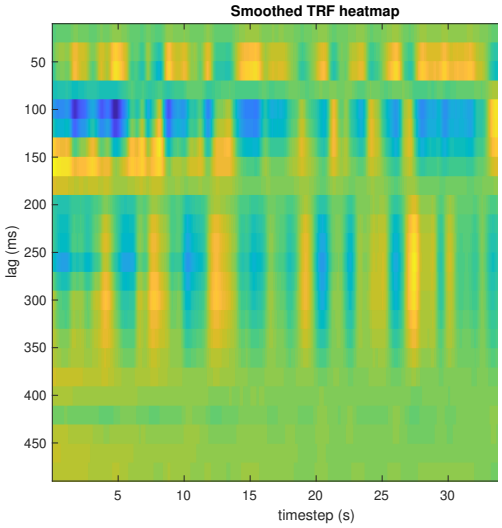


(a) Convergence of n_1.

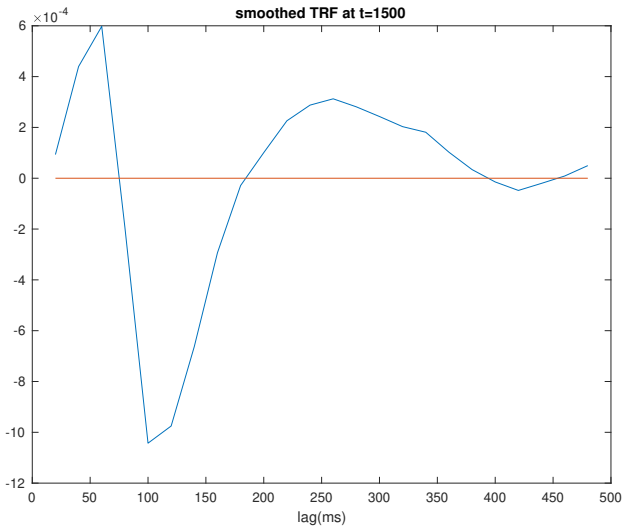


(b) Convergence of n_2.

Figure 4.5: Convergence of weights.



(a) Heatmap of smoothed TRF with estimated parameters.



(b) Smoothed TRF at timestep $t = 1500$

Figure 4.6: The smoothed TRF with estimated parameters.

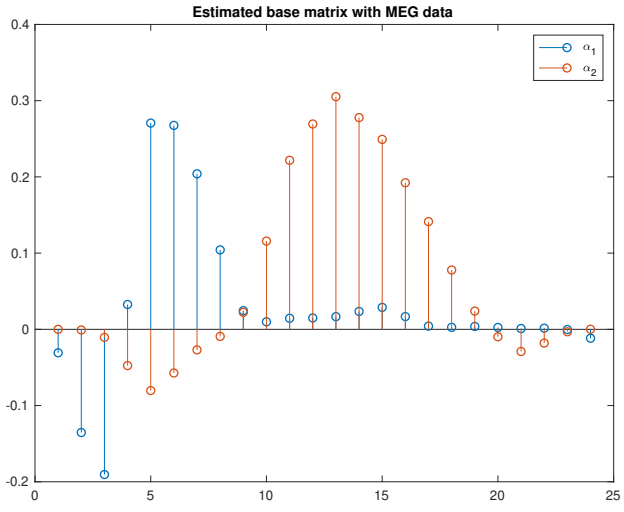


Figure 4.7: Estimated base matrix with acquired MEG.

Bibliography

[1] Yunchuan Li and Michael Fu. Sequential first-order response surface methodology augmented direct gradient. In *Proceedings of the 2018 Winter Simulation Conference*, pages 2143–2154. IEEE, 2018.

[2] Yunchuan Li and Michael Fu. Direct gradient augmented response surface methodology as stochastic approximation. *Operations Research*, 2020. Under review.

[3] Yunchuan Li, Michael C. Fu, and Jie Xu. Monte Carlo tree search with optimal computing budget allocation. In *Proceedings of 58th IEEE Conference on Decision and Control*, pages 6332–6337 vol.3, Dec 2019.

[4] Yunchuan Li, Michael C. Fu, and Jie Xu. Monte Carlo tree search with optimal computing budget allocation. *IEEE Transactions on Automatic Control*, 2020. Submitted.

[5] Yunchuan Li, Behtash Babadi, Michael C. Fu, Steven Marcus, and Johnathan Simon. An expectation-maximization algorithm for state-space model with autoregressive process noise. 2020. In preparation.

[6] Yu Chi Ho and Xiren Cao. Perturbation analysis and optimization of queueing networks. *Journal of Optimization Theory and Applications*, 40(4):559–582, 1983.

[7] Paul Glasserman. *Gradient Estimation via Perturbation Analysis*. Springer, 1991.

[8] Reuven Y Rubinstein and Alexander Shapiro. *Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. John Wiley & Sons Inc, 1993.

[9] Michael C. Fu. Optimization via simulation: A review. *Annals of Operations Research*, 53(1):199–247, Dec 1994.

[10] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[11] Julius R. Blum. Multidimensional stochastic approximation methods. *Ann. Math. Statist.*, 25(4):737–744, 12 1954.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[14] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.

[15] Marie Chau, Huashuai Qu, and Michael C Fu. A new hybrid stochastic approximation algorithm. In *Proceedings of the 12th International Workshop on Discrete Event Systems (WODES)*, pages 241–246. Elsevier, 2014.

[16] M. Chau, M. C. Fu, H. Qu, and I. O. Ryzhov. Simulation optimization: A tutorial overview and recent developments in gradient-based methods. In *Proceedings of the 2014 Winter Simulation Conference*, pages 21–35, 2014.

[17] M. Chau, M.C. Fu, J.J. Lee, and H. Qu. Multivariate stochastic approximation using a secant-tangents averaged (star) gradient. under review, 2018.

[18] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(1):1–38, 1951.

[19] R. Mead and D. J. Pike. A biometrics invited paper. a review of response surface methodology from a biometric viewpoint. *Biometrics*, 31(4):803–851, 1975.

[20] Shien-Ming Wu. Tool-life testing by response surface methodology: Part 1. *Journal of Engineering for Industry*, 86(2):105–110, 1964.

[21] Jenn-Tsong Horng, Nun-Ming Liu, and Ko-Ta Chiang. Investigating the machinability evaluation of hadfield steel in the hard turning with al2o3/tic mixed ceramic tool based on the response surface methodology. *Journal of Materials Processing Technology*, 208(1):532 – 541, 2008.

[22] F. Keyzer, J. Kleijnen, E. Mullenders, and A. van Reeken. Optimization of priority class queues, with a computer center case study. *American Journal of Mathematical and Management Sciences*, 1(4):341–358, 1981.

[23] Y. Carson and A. Maria. Simulation optimization: Methods and applications. In *Proceedings of the 1997 Winter Simulation Conference*, pages 118–126, 12 1997.

[24] Russell R. Barton and Martin Meckesheimer. Metamodel-based simulation optimization. volume 13 of *Handbooks in Operations Research and Management Science*, pages 535 – 574. Elsevier, 2006.

[25] Thomas Bartz-Beielstein and Mike Preuss. Experimental research in evolutionary computation. In *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '07, pages 3001–3020, New York, NY, USA, 2007. ACM.

[26] Averill M Law, W David Kelton, and W David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill New York, 2007.

[27] Raymond H. Myers, André I. Khuri, and Walter H. Carter. Response surface methodology: 1966-1988. *Technometrics*, 31(2):137–157, 1989.

[28] Jack P. C. Kleijnen. Response surface methodology. In Michael C Fu, editor, *Handbook of Simulation Optimization*, pages 81–104. Springer New York, New York, NY, 2015.

[29] Robin L Plackett and J Peter Burman. The design of optimum multifactorial experiments. *Biometrika*, 33(4):305–325, 1946.

[30] Genichi Taguchi. Introduction to quality engineering: Designing quality into products and processes. Technical report, 1986.

[31] George EP Box and Norman R Draper. Robust designs. *Biometrika*, 62(2):347–352, 1975.

[32] Whasoo B. Kim and Norman R. Draper. Choosing a design for straight line fits to two correlated responses. *Statistica Sinica*, 4(1):275–280, 1994.

[33] Olaf Krafft and Martin Schaefer. D-optimal designs for a multivariate regression model. *Journal of Multivariate Analysis*, 42(1):130–140, 1992.

[34] Michael C Fu and Huashuai Qu. Regression models augmented with direct stochastic gradient estimators. *INFORMS Journal on Computing*, 26(3):484–499, 2014.

[35] Guillermo Miró-Quesada and Enrique Del Castillo. An enhanced recursive stopping rule for steepest ascent searches in response surface methodology. *Communications in Statistics-Simulation and Computation*, 33(1):201–228, 2004.

[36] Suntara Cahya, Enrique Del Castillo, and John J Peterson. Computation of confidence regions for optimal factor levels in constrained response surface problems. *Journal of Computational and Graphical Statistics*, 13(2):499–518, 2004.

[37] Kuo-Hao Chang, L Jeff Hong, and Hong Wan. Stochastic trust region gradient-free method (strong)-a new response-surface-based algorithm in simulation optimization. In *Proceedings of the 1997 Winter Simulation Conference*, pages 346–354. IEEE, 2007.

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

[39] Harold Joseph Kushner and Dean S Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, volume 26. Springer Science & Business Media, 1978.

[40] Q. Wang and M. Ye. Rate of convergence analysis of simultaneous perturbation stochastic approximation algorithm for time-varying loss function. In *2014 American Control Conference*, pages 5192–5197, 2014.

[41] Raghu Pasupathy, Peter Glynn, Soumyadip Ghosh, and Fatemeh Hashemi. On sampling rates in simulation-based recursions. *SIAM Journal on Optimization*, 28:45–73, 01 2018.

[42] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[43] Hyeong Soo Chang, Michael C Fu, Jiaqiao Hu, and Steven I Marcus. An adaptive sampling algorithm for solving Markov decision processes. *Operations Research*, 53(1):126–139, 2005.

[44] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[45] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.

[46] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[47] Philip Hingston and Martin Masek. Experiments with Monte Carlo othello. In *2007 IEEE Congress on Evolutionary Computation*, pages 4059–4064, Sept 2007.

[48] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[49] Pierre-Arnaud Coquelin and Rémi Munos. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

[50] Maarten Peter Dirk Schadd. *Selective search in games of different complexity*. PhD thesis, Maastricht University, 2011.

[51] Kazuki Teraoka, Kohei Hatano, and Eiji Takimoto. Efficient sampling method for Monte Carlo tree search problem. *IEICE TRANSACTIONS on Information and Systems*, 97(3):392–398, 2014.

[52] Emilie Kaufmann and Wouter M Koolen. Monte-Carlo tree search by best arm identification. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4897–4906. Curran Associates, Inc., 2017.

[53] Jean-Bastien Grill, Michal Valko, and Remi Munos. Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4680–4688. Curran Associates, Inc., 2016.

[54] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, March 1985.

[55] Chun-Hung Chen, Jianwu Lin, Enver Yücesan, and Stephen E Chick. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 10(3):251–270, 2000.

[56] Chun-Hung Chen. An effective approach to smartly allocate computing budget for discrete event simulation. In *Proceedings of 34th IEEE Conference on Decision and Control*, volume 3, pages 2598–2603 vol.3, Dec 1995.

[57] Loo Hay Lee, Ek Peng Chew, Suyan Teng, and David Goldsman. Optimal computing budget allocation for multi-objective simulation models. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, volume 1, page 594, Dec 2004.

[58] Chun-Hung Chen, Donghai He, Michael Fu, and Loo Hay Lee. Efficient simulation budget allocation for selecting an optimal subset. *INFORMS Journal on Computing*, 20(4):579–595, 2008.

[59] Si Zhang, Loo Hay Lee, Ek Peng Chew, Jie Xu, and Chun-Hung Chen. A simulation budget allocation procedure for enhancing the efficiency of optimal subset selection. *IEEE Transactions on Automatic Control*, 61(1):62–75, Jan 2016.

[60] Alexander Shleyfman, Antonín Komenda, and Carmel Domshlak. On interruptible pure exploration in multi-armed bandits. In *AAAI Conference on Artificial Intelligence*, 2015.

[61] Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 379–387. Curran Associates, Inc., 2014.

[62] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 23–37, Berlin, Heidelberg, 2009. Springer.

[63] Chun-Hung Chen and Loo Hay Lee. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1st edition, 2010.

[64] Morris H DeGroot. *Optimal Statistical Decisions*, volume 82. John Wiley & Sons, 2005.

[65] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[66] Chun-Hung Chen, Donghai He, and Michael Fu. Efficient dynamic simulation allocation in ordinal optimization. *IEEE Transactions on Automatic Control*, 51(12):2005–2009, 2006.

[67] Daniel R. Jiang, Lina Al-Kanj, and Warren B. Powell. Monte Carlo tree search with sampled information relaxation dual bounds, 2017.

[68] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[69] A. M. H. J. Aertsen, J. H. J. Olders, and P. I. M. Johannesma. Spectro-temporal receptive fields of auditory neurons in the grassfrog. *Biological Cybernetics*, 39(3):195–209, 1981.

[70] J. J. Eggermont, P. I. M. Johannesma, and A. M. H. J. Aertsen. Reverse-correlation methods in auditory research. *Quarterly Reviews of Biophysics*, 16(3):341–414, 1983.

[71] Frédéric Theunissen, Stephen David, Nandini Chatterjee, Anne Hsu, W Vinje, and Jack Gallant. Estimating spatio-temporal receptive fields of auditory and visual neurons from their responses to natural stimuli. *Network (Bristol, England)*, 12:289–316, 09 2001.

[72] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Statist.*, 28(2):337–407, 04 2000.

[73] Jerome Friedman, Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. Discussion of boosting papers. *Ann. Statist*, 32:102–107, 2004.

[74] Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. *Ann. Statist.*, 33(4):1538–1579, 08 2005.

[75] Nai Ding and Jonathan Z. Simon. Emergence of neural encoding of auditory objects while listening to competing speakers. *Proceedings of the National Academy of Sciences*, 109(29):11854–11859, 2012.

[76] Jonathan Fritz, Shihab Shamma, Mounya Elhilali, and David Klein. Rapid task-related plasticity of spectrotemporal receptive fields in primary auditory cortex. *Nature Neuroscience*, 6(11):1216–1223, 2003.

[77] Jonathan Fritz, Mounya Elhilali, and Shihab Shamma. Active listening: Task-dependent plasticity of spectrotemporal receptive fields in primary auditory cortex. *Hearing Research*, 206(1):159 – 176, 2005. 3rd Symp. on Molec. Mechanisms in Central Auditory Function, Plasticity and Disorder.

[78] Christoph E. Schreiner and Jeffery A. Winer. Auditory cortex mapmaking: Principles, projections, and plasticity. *Neuron*, 56(2):356 – 365, 2007.

[79] Serin Atiani, Mounya Elhilali, Stephen David, Jonathan Fritz, and Shihab Shamma. Task difficulty and performance induce diverse adaptive patterns in gain and shape of primary auditory cortical receptive fields. *Neuron*, 61:467–80, 03 2009.

[80] Jyrki Ahveninen, Matti Hämäläinen, Iiro Jääskeläinen, Seppo Ahlfors, Samantha Huang, Fa-Hsuan Lin, Tommi Raij, Mikko Sams, Christos Vasios, and John Belliveau. Attention-driven auditory cortex short-term plasticity helps segregate relevant sounds from noise. *Proceedings of the National Academy of Sciences of the United States of America*, 108:4182–7, 02 2011.

[81] Nima Mesgarani and Edward F. Chang. Selective cortical representation of attended speaker in multi-talker speech perception. *Nature*, 485(7397):233–236, 2012.

[82] Edmund C. Lalor and John J. Foxe. Neural responses to uninterrupted natural speech can be extracted with precise temporal resolution. *European Journal of Neuroscience*, 31(1):189–193, 2010.

[83] W. Wu, J. E. Kulkarni, N. G. Hatsopoulos, and L. Paninski. Neural decoding of hand motion using a linear state-space model with hidden states. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(4):370–378, 2009.

[84] Alireza Sheikhattar, Sina Miran, Ji Liu, Jonathan B. Fritz, Shihab A. Shamma, Patrick O. Kanold, and Behtash Babadi. Extracting neuronal functional network dynamics via adaptive granger causality analysis. *Proceedings of the National Academy of Sciences*, 115(17):E3869–E3878, 2018.

[85] Nai Ding. Neural coding of continuous speech in auditory cortex during monaural and dichotic listening. *Journal of Neurophysiology*, 107:78–89, 01 2012.

[86] Nai Ding and Jonathan Z. Simon. Adaptive temporal encoding leads to a background-insensitive cortical representation of speech. *The Journal of Neuroscience : the Official Journal of the Society for Neuroscience*, 33 13:5728–35, 2013.

[87] Nai Ding and Jonathan Z. Simon. Robust cortical encoding of slow temporal modulations of speech. In Brian C. J. Moore, Roy D. Patterson, Ian M. Winter, Robert P. Carlyon, and Hedwig E Gockel, editors, *Basic Aspects of Hearing*, pages 373–381, New York, NY, 2013. Springer New York.

[88] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer International Publishing, 4 edition, 2017.

[89] Anne C. Smith and Emery N. Brown. Estimating a state-space model from point process observations. *Neural Comput.*, 15(5):965–991, May 2003.

[90] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[91] Mark Zlochin, Mauro Birattari, Nicolas Meuleau, and Marco Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1):373–395, Oct 2004.

[92] Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89 – 112, 1997.

[93] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2004.

[94] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*, 1991.

[95] Jiaqiao Hu, Michael C. Fu, and Steven I. Marcus. A model reference adaptive search method for global optimization. *Operations Research*, 55(3):549–568, 2007.