UNIVERSITY OF CALIFORNIA
RIVERSIDE

Wireless Relay Transceiver Design, Thermal Modeling and Fast Subspace Tracking

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Shengyang Xu

December 2012

Dissertation Committee:

Dr. Yingbo Hua, Chairperson
Dr. Sheldon X.-D Tan
Dr. Ertem Tuncel

The Dissertation of Shengyang Xu is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to express my deepest gratitude to my advisor, Professor Yingbo Hua, without whose intellectual guidance I would never have reached this stage of my academic life. I benefit a lot from his incredibly profound knowledge and critical thinking about significant problems. His research approaches and attitudes greatly influence me throughout my academic research.

I am also honored to have Professor Sheldon X.-D Tan and Professor Ertem Tuncel as my committee members. Professor Tan also introduced me to the topic of thermal modeling and gave me a lot of invaluable suggestions. Professor Ilya Dumer and Professor Jay A. Farrell served in the committee of my oral qualifying exam and provided many useful advice about my research proposal and how my research should continue.

It is my pleasure to be a member of Professor Hua's lab where all the lab members are all so intelligent and have good personalities. Dr. Kezhu Hong helped me a lot at the beginning stage of my Ph.D. study, in both research and everyday life. I learned a lot from senior members Dr. Yue Rong, Yuan Yu, Yi Huang and Ting Kong as they were all so patient and efficient to explain to me difficult concepts and ideas. Xiang Dong and Jie Liang went through almost every course with me during the first two years of my graduate study. I also enjoyed every moment together with Haitao Liu, Qian Gao, Ali Cirik, Yiming Ma, Armen Gholian.

On my research of thermal modeling, Dr. Hai Wang, Dr. Ruijing Shen and Dr. Duo Li were always enthusiastic to answer my questions and helped me get a deeper understanding of every aspect of the topic.

During my graduate study, I would not have achieved much without the help and encouragement of my dear friends, Guanxiong Liu, Haipeng Ding, Tianhui Wang, Wenyu Huo, Peihui Zhang, Jifei Ban, Yang Gao, Leijie Wang, Quefeng He, Yunfan Li, Licheng Luo, He Tang, Jin Li, Li Wang, Qiang Fang, Songfan Yang, Hongjia Li, Wuyang Dai, Chenxi Zhang, Yunfei Shi, Chen Wang, Chao Shi, Yan Han, Yunsheng Wang, Zhiyu Yang, Bob Zhao, Lei Dong, Feng Xu, Robin Tu, Qi Song, Michelle Qi, Geyuan Liu and many others. I appreciate all your supports through my entire study.

Finally, not much would have been possible in my life without the love, encouragement and unconditional support I receive from my parents, my mother Huixin Yue and my father Changnian Xu. To my parents, I dedicate this dissertation.

To my parents.

ABSTRACT OF THE DISSERTATION

Wireless Relay Transceiver Design, Thermal Modeling and Fast Subspace Tracking

by

Shengyang Xu

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2012
Dr. Yingbo Hua, Chairperson

Wireless relays are such methods that can be rapidly deployed to enhance the coverage, reliability and throughput of a wireless network subject to power and spectral constraints. When equipped with multiple antennas, Multiple Input Multiple Output (MIMO) relays, are particularly useful for scattering rich and non-line-of-sight environment. The first part of this dissertation considers a system where two users exchange information via a non-regenerative half-duplex two-way MIMO relay. We study the transceiver design including both source covariance matrices at the two users and beamforming matrix at the relay to maximize the achievable weighted sum rate of the system. We compare the convergence behaviors of the proposed algorithms and demonstrate their advantages over prior algorithms. We also show an optimal structure of the relay matrix, which is useful to reduce the search complexity.

As advanced architectures and high-performance hardware are required to implement more powerful but complicated algorithms such as those in a two-way relay system, multiple cores are often integrated on a chip of shrinking size. However, the correspond-

ing dramatically increased power density may lead to significant adverse effects. Dynamic thermal management is widely used to mitigate this problem, where thermal modeling and temperature prediction play the key roles. Unlike conventional bottom-up approaches, a Linear Time-Invariant (LTI) Multiple-Input-Multiple-Output (MIMO) black-box model is adopted and Least Square (LS) based model averaging algorithm with model screening is developed with less temperature prediction errors than the traditional LS algorithm based on model order selection.

For VLSI thermal modeling, Model Order Reduction (MOR) is an efficient technique to reduce the modeling complexity where subspace-based methods could be successfully applied. In the third part, motivated by MOR for thermal modeling, subspace tracking is investigated as one of the key procedures for subspace-based methods. We explain the reason to enlarge the actual tracking dimension and equip bi-iteration SVD algorithm with multiple inner iterations and Ritz acceleration. Our proposed algorithms are demonstrated to have much improved performance of both convergence rate and tracking accuracy compared to existing algorithms while still keeping linear complexity without many additional computational consumptions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The radio frequency spectrum for mobile wireless communication is becoming increasingly crowded especially in large cities. Developing technologies for efficient spectral usage is becoming more important. Wireless relays are such methods that can be rapidly deployed to enhance the coverage, reliability and throughput of a wireless network subject to power and spectral constraints [14] [52] [50] [51] [26]. Wireless relays equipped with multiple antennas, also called Multiple Input Multiple Output (MIMO) relays [66] [59] [6] [49], are particularly useful for scattering rich and non-line-of-sight environment.

Since advanced architectures are required to implement more complicated algorithms such as in a relay system, more high-performance cores are integrated on a chip of shrinking size in multicore microprocessor architecture and its power density is expected to increase drastically, which could lead to rapidly increasing chip temperatures and associated significant adverse effects on chip packaging and cooling costs, circuit reliability, leakage power and system performance, e.g., see [10] and [54]. To alleviate such problems,

much attention is needed for dynamic thermal management where hardware and/or software techniques are used to lower the temperatures once a predicted future temperature is higher than a threshold. For dynamic thermal management, accurate temperature prediction plays a key role or otherwise the system could suffer from reliability or performance degradation. A critical component needed for dynamic thermal management is thermal modeling which describes the relationship between input powers and output temperatures.

For VLSI thermal modeling, subspace-based methods could be applied to Model Order Reduction (MOR) as an efficient technique to reduce the modeling complexity. They have also been successfully used to a wide range of signal processing applications, such as parameter estimation, source localization, and adaptive filtering [69]. The span of a sequence of signal vectors is divided into a principal subspace and its complementary subspace, a minor subspace, that serve as the keys for further processing in those algorithms. In order to estimate the principal subspace, Singular Value Decomposition (SVD) of a data matrix obtained from the signal vectors is commonly used but leads to high computational costs although providing the full information of the signal span. Moreover, in adaptive applications where new signal vector is obtained at each time instant, tracking the corresponding varying subspace imposes a tighter limitation on the algorithm computational complexity. Therefore, more efficient algorithms are desired for fast tracking the subspace of an updating signal vector sequence in the field of adaptive signal processing.

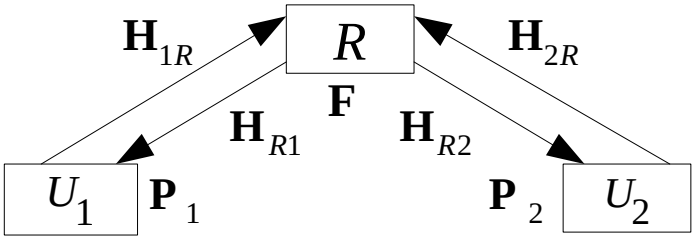

Figure 1.1: A non-regenerative two-way MIMO relay system where the two users are denoted by U_1 and U_2 respectively, and the relay node by R.

1.1 Non-Regenerative Two-Way MIMO Relay System

The idea of two-way relay has been studied in [47], [20], [34], [65], [37], [77], [62]. Central to this idea is that two users served by a relay can transmit their information to the relay simultaneously in one channel and the relay forwards a combined information to both users in another channel. And since each user knows its own information, it can remove its self-interference from the signal received from the relay provided that the required channel state information is available. Although the relay is half-duplex, i.e., requiring two channels to relay a signal, the two-way scheme allows two users to share both channels concurrently. This leads to a high spectral efficiency.

The authors of [47] appear to be the first who proposed the idea of two-way relay, where they considered a system with a single-antenna relay and single-antenna users. The work [20] studied coding schemes for a regenerative MIMO relay system. In [34], several two-way relay schemes were proposed and their capacity regions were explored. Insights were obtained for single-antenna users and single-antenna relay. In [65], the authors considered a power minimization problem for single-antenna users and a non-regenerative MIMO relay. For multi-antenna users, they proposed a heuristic sub-optimal method called dual channel

matching strategy. The work [37] considered the maximization of the sum rates of two single-antenna users assisted by a non-regenerative MIMO relay under a high SNR assumption. In [77], a convex optimization algorithm was formulated to compute the capacity region of a relay system same as in [37] but without the high SNR assumption. The algorithm developed in [77] is however not applicable for multi-antenna users. In [62], suboptimal power allocation methods were proposed to maximize the system throughput of a non-regenerative MIMO relay system.

For this topic we consider the problem of jointly designing/computing the spatial source covariance matrices (source matrices) and the spatial relay transformation matrix (relay matrix) for a non-regenerative two-way MIMO relay system where all nodes (i.e., the relay and the two users) have multiple antennas. In practice, one of the two "users" can be an access point, and the other a user equipment. In many situations, both users are already equipped with multiple antennas. When a relay is placed between them to improve the power and spectral efficiency, the relay should be a MIMO relay and hence each of the nodes in the system is a MIMO node. It is not hard to imagine that the source matrix (also called transmit covariance matrix) used at each user and the relay matrix used at the relay all affect the system capacity. It is therefore important to understand how to optimally design these matrices and how much capacity gain such a design can yield. This problem has not received enough attention from researchers.

It is known that a two-user (half-duplex) MIMO relay system as in Fig. 1.1 can be treated as *two* one-way (half-duplex) MIMO relay systems, and each one-way MIMO relay system can be treated as in [49]. Partially due to the known optimal structure of the

source and relay matrices as revealed in [49], the complexity of designing the source and relay matrices for the one-way scheme is much lower than that for the two-way scheme. But for two users to exchange their information via a half-duplex relay, the one-way scheme has a factor-4 loss of spectral efficiency while the two-way scheme has only a factor-2 loss. Therefore, the two-way scheme is approximately twice as spectrally efficient as the one-way scheme.

1.2 Thermal Modeling and Temperature Prediction

Thermal modeling can be done via bottom-up approaches or black-box approaches. The bottom-up approaches are based on the physical properties of the circuits and devices in a microprocessor, e.g., see [54], [75], and [30]. For accurate thermal modeling of a very large scale system, the bottom-up approaches can be too costly to be practical. For this reason, the black-box approaches, which generate thermal models only based on the observed external signals such as powers and temperatures, have attracted much interest recently. The black-box approaches have been recently applied in [13], [38], [16].

In [13], the authors use an ARMA (autoregressive and moving average) thermal model and assume that the running workload is a stationary stochastic process. They also use a hypothesis detection to update the model when the running workload on the cores changes. However, the unreliability of the predicted temperatures between two updated models threats the reliability and performance of the system. Also, the assumption of statistically stationary running workload for a sufficiently long time to allow a reliable statistical modeling is hard to meet in practice. In [38], the authors considered a black-

box MIMO (multi-input multi-output) linear time-invariant (LTI) thermal model which treats the powers to all cores as the system inputs and the temperatures from all cores as the system outputs. The thermal characteristic of a multicore system is captured by the impulse response matrix resulting from a modified generalized pencil of function method also called ThermPOF. For this method, only step input functions are allowed to train the model, and only one core can be excited at a time (i.e., keeping inputs to all other cores at zero). This requirement makes ThermPOF only useful off-line. In [16], the authors considered a state-space thermal modeling method called ThermSID which does not have the limitation for ThermPOF.

1.3 Fast Subspace Tracking Algorithms

Assume that the dimension of the desired principal subspace r is much smaller than the dimension of the signal vectors N. In terms of several criteria, subspace tracking algorithm could be grouped into different categories. From the point of view of mathematical tools used to track the principal subspace, we have batch/adaptive SVD method such as QR algorithm, Jacobi rotation, power iteration and Lanczos method [45] [61] [41] [70] [71] as the first category. In the second category, variations of rank-one updating algorithm such as [31] [15] were proposed. The third category of algorithms regards the SVD as a constrained or unconstrained optimization problem and capitalizes gradient based methods, Gauss-Newton iterations and conjugate gradient methods to estimate the principal or minor subspace adaptively [32] [43] [63] [74] [68] [5] []. Other algorithms such as rank revealing decomposition have also been proposed [56] [7] [27] [4] [21]. The subspace tracking

algorithms can also be distinguished by the algorithm complexity for each data update. Using the *flop* defined as a complex multiplication and a complex addition to evaluate the algorithm complexity, we can classify them as *high complexity* with $O(N^2r)$ flops, *medium complexity* with $O(Nr^2)$ flops and *low complexity* with $O(Nr)$ flops. Note that although minor subspace also has important applications [67] [42], the tracking of principal subspace is the focus of this dissertation.

Next, we briefly describe some of the key algorithms from many subspace tracking algorithms in order to show how our proposed work relates to the existing algorithms. Since $N \gg r$, the category of high complexity algorithms with complexity of $O(N^2r)$ flops such as Lanczos-type algorithms proposed in [71] could still be too costly so that lower complexity algorithms of *linear complexity* of $O(Nr^2)$ or $O(Nr)$ flops as linear functions of N attract most of the attentions, where a *flop* is defined as a complex multiplication and a complex addition. Thus, we only focus on this category of subspace tracking algorithms with linear complexity. Among the existing linear complexity algorithms, Karasalo's algorithm [31] only requires the SVD of a smaller dimensional matrix and has the complexity of $O(Nr^2)$. By interpreting the principal subspace as the solution of a projection like unconstrained minimization problem, Projection Approximation Subspace Tracking (PAST) algorithm [73] applies recursive least square techniques to develop a fast tracking algorithm with complexity $O(Nr)$. However, the orthonormality of the estimated subspace matrix is not guaranteed for PAST but could be explored by OPAST algorithm with an explicit orthonormalization at each data update based on PAST [1]. Alternatively, Novel Information Criterion (NIC) algorithm [40] also ensures the orthonormality of the subspace matrix

by introducing the leakage factor as a generalization of PAST. It could be shown in [29] that PAST algorithm and its extensions are essentially based on power/orthogonal iteration and can be formulated under a general framework. In terms of classical bi-iteration SVD method [12] [55], bi-iteration SVD (Bi-SVD) subspace tracking algorithms [58] are proposed with excellent convergence properties where a lower-rank consistent approximation replaces the actual data matrix at each data update. Similar to [58], Sliding Window Adaptive SVD (SWASVD) algorithm [3] employs a different lower-rank consistent approximation but use the rectangular sliding window data matrix instead of exponential forgetting window in order to provide more robust tracking of abrupt signal changes. Both Bi-SVD and SWASVD have the complexity of $O(Nr^2)$, which could be further reduced to $O(Nr)$ by appropriate approximations as their ultrafast versions with little performance penalty. Furthermore, more efficient subspace tracking algorithm with complexity $O(Nr)$, such as Bi-Iterative Least Square (Bi-LS) algorithm and Fast Approximated Power Iteration algorithm, were studied in [44] [2].

1.4 Contributions

In this dissertation, efficient algorithms for computing the source matrices and the relay matrix that maximize a (weighted) sum rate are developed in Chapter 2. Dealing with a non-convex optimization problem we optimize the source matrices and the relay matrix alternately until convergence. When fixed relay matrix is considered, the problem of finding the optimal source matrices for the two multi-antenna users is convex and the solution can be obtained by the generalized water-filling algorithm developed in [76] for uniformly weighted

sum rate. When we assume that the source matrices are fixed, we propose a hybrid gradient algorithm and an iterative weighted minimum mean square error (WMMSE) algorithm to compute the best relay matrix. The hybrid gradient algorithm combines both the gradient descent search and the Newton's search while the iterative WMMSE algorithm is based on an alternate convex search of the relay matrix and the MMSE equalizer at each user. The optimal structure of the relay matrix is also shown and useful to reduce the computational complexity when the number of antennas at the relay is more than twice the number of the antennas at each user. We will show that our search algorithms for the relay matrix produce the maximum sum rate much faster than existing methods when applied to the case of single-antenna users.

We present a simple and perhaps the most basic approach to Black-Box MIMO Thermal Modeling. We treat a multicore microprocessor system as a deterministic MIMO thermal system whose inputs are the powers consumed by the cores in the system and whose outputs are the temperatures at certain predetermined points in the system. The input-output relationship is simply governed by a classical deterministic MIMO ARMA model. The parameters of the deterministic MIMO ARMA model can be determined by using the classical Least Square (LS) with model order selection where its uncertainty is neglected after the final model is chosen [22]. To overcome the uncertainty inherited in model order selection, a model averaging algorithm based on least square is proposed with model screening. We show that this algorithm can provide better prediction performance compared with model order selection.

For subspace tracking, in order to further improve the algorithm performance,

multiple inner iterations could be applied to an approximated data matrix at each data update based on classical bi-iteration SVD-based algorithms, intuitively. However, in Chapter 4, we follow this idea and find that the performance improvement is just marginal no matter how many inner iterations are operated. This could be explained by the fact that during the transition period there are actually two sets of principal subspaces included in the data matrix. Thus, the actual dimension of the underlying principal subspace is doubled while only a half of the actual dimension is contained in the approximated data matrix. The inaccuracy of this approximation matrix leads to the desire of tracking a larger dimension of principal subspace. Meanwhile, multiple inner iterations at each data update also inevitably increase the algorithm complexity. Fortunately, Ritz acceleration can be employed to further improve the convergence rate and tracking accuracy of the proposed algorithm with a much smaller number of inner iterations, where the SVD of a much smaller-dimensional matrix is computed when $N \gg r$. As existing algorithms cannot improve the convergence rate and tracking accuracy any more, increasing the algorithm complexity turns out to be the only option to achieve such a target. Under such a premise the proposed algorithms with multiple inner iterations and Ritz acceleration are demonstrated to accomplish much better performance without algorithm complexity greatly increased.

Chapter 2

Optimal Design of Source-and-Relay Matrices for a Non-Regenerative Two-Way MIMO Relay System

2.1 Introduction

For a two-way MIMO relay system, we will present efficient algorithms for computing the source matrices and the relay matrix that maximize a (weighted) sum rate in Chapter 2. Our algorithms are based on an alternate optimization approach where the source matrices and the relay matrix are optimized alternately until convergence. When the relay matrix is fixed, finding the optimal source matrices for the two multi-antenna

users is a convex problem, and for uniformly weighted sum rate, it can be solved by the generalized water-filling algorithm developed in [76]. When the source matrices are fixed, we develop a hybrid gradient algorithm and an iterative weighted minimum mean square error (WMMSE) algorithm to find the best relay matrix. The hybrid gradient algorithm combines the gradient descent search and the Newton's search. The iterative WMMSE algorithm is based on an alternate convex search of the relay matrix and the MMSE equalizer at each user. This idea was inspired by the work [11] on source precoding for MIMO broadcasting. We will also show an optimal structure of the relay matrix, which is useful to reduce the computational complexity when the number of antennas at the relay is more than twice the number of the antennas at each user. This result is a generalization of one shown in [77] from single-antenna users to multi-antenna users. Furthermore, we will demonstrate that when applied to the case of single-antenna users, our search algorithms for the relay matrix yield the maximum sum rate much faster than the convex method developed in [77].

It is important to note that while the idea of gradient search is well known, its application to the problem addressed here is the first. The problem structure is also exploited to simplify the expression and computation of the gradient vector and Hessian matrix. The resulting algorithm provides a new and useful perspective of other alternative algorithms such as one in [77] and WMMSE. This study reveals a capacity potential of a two-way MIMO relay system that no other existing work has been able to reveal.

The rest of this chapter is organized as follows. In Section 2.2, we describe in more details the non-regenerative two-way MIMO relay system, and formulate the optimization problems we aim to solve. In Section 2.3, we present an optimal structure of the relay matrix

under the condition that the number of antennas at the relay is larger than twice of that at each of the users. In 2.4, we present the hybrid gradient method to compute the optimal relay matrix with fixed source covariance matrices. In Section 2.5, the iterative WMMSE method is presented. In Section 2.6, the procedure of finding the optimal source covariance matrices with fixed relay matrix is described. Section 2.7 illustrates the performance of our algorithms. The conclusion is given in Section 2.8.

2.2 Problem Formulation

The two-way relay system under consideration is illustrated in Fig. 1.1, which involves two users and one relay. The two users are denoted by U_1 and U_2 each with N antennas. Since the two users exchange information with each other, they are also referred to as sources and destinations. The relay is denoted by R with M antennas. We focus on a single carrier with a narrow enough bandwidth so that the channels between nodes are flat fading.

The two-way relay scheme has two phases. In the first phase, U_1 and U_2 transmit $\mathbf{x}_1 = \mathbf{P}_1\mathbf{s}_1$ and $\mathbf{x}_2 = \mathbf{P}_2\mathbf{s}_2$, respectively. Here, each of $\mathbf{s}_1$ and $\mathbf{s}_2$ consists of N independent complex symbols, $\mathbf{s}_i \in \mathbb{C}^{N\times 1}$ and $\mathbf{E}[\mathbf{s}_i\mathbf{s}_i^H] = \mathbf{I}$, $i = 1, 2$ and $\mathbf{P}_1$ and $\mathbf{P}_2$ are two square precoding matrices. The signal received at R can be expressed as

$$\mathbf{y}_R = \mathbf{H}_{1R}\mathbf{x}_1 + \mathbf{H}_{2R}\mathbf{x}_2 + \mathbf{n}_R \tag{2.1}$$

where $\mathbf{H}_{iR} \in \mathbb{C}^{M\times N}$, $i = 1, 2$, denotes the channel matrices (channel state information)

from U_i to R, and $\mathbf{n}_R \in \mathbb{C}^{M\times 1}$ is the noise. We assume that the noise is complex white Gaussian, i.e., $\mathbf{n}_R \sim \mathcal{CN}(0, \mathbf{I})$.

In the second phase, R transmits $\mathbf{y}'_R = \mathbf{F}\mathbf{y}_R$ where $\mathbf{F} \in \mathbb{C}^{M\times M}$ is the relay transformation matrix. The transmit power consumed at the relay is

$$p_R = \mathrm{Tr}\left(\mathbf{F}\mathbf{H}_{1R}\mathbf{R}_{x_1}\mathbf{H}_{1R}^H\mathbf{F}^H + \mathbf{F}\mathbf{H}_{2R}\mathbf{R}_{x_2}\mathbf{H}_{2R}^H\mathbf{F}^H + \mathbf{F}\mathbf{F}^H\right) \tag{2.2}$$

where $\mathbf{R}_{x_i} = \mathbf{E}[\mathbf{x}_i\mathbf{x}_i^H] = \mathbf{P}_i\mathbf{P}_i^H$, $i = 1, 2$, denotes the source covariance matrix at U_i. Note that $\mathbf{P}_i$ is a matrix square root of $\mathbf{R}_{x_i}$. The signals received at U_i, $i = 1, 2$, are

$$\mathbf{y}_i = \mathbf{H}_{Ri}\mathbf{F}\mathbf{H}_{iR}\mathbf{x}_i + \mathbf{H}_{Ri}\mathbf{F}\mathbf{H}_{\bar{i}R}\mathbf{x}_{\bar{i}} + \mathbf{H}_{Ri}\mathbf{F}\mathbf{n}_R + \mathbf{n}_i \tag{2.3}$$

where $\mathbf{H}_{Ri} \in \mathbb{C}^{N\times M}$ denote the channel matrices from R to U_i, and $\mathbf{n}_i \in \mathbb{C}^{N\times 1}$ is the noise assumed to be $\mathcal{CN}(0, \mathbf{I})$. Here, $\bar{i} = 2$ for $i = 1$, and $\bar{i} = 1$ for $i = 2$. We see that $\mathbf{H}_{R1}\mathbf{F}\mathbf{H}_{1R}\mathbf{x}_1$ is the self-interference at U_1, and $\mathbf{H}_{R2}\mathbf{F}\mathbf{H}_{2R}\mathbf{x}_2$ is the self-interference at U_2. With a perfect knowledge of $\mathbf{H}_{R1}\mathbf{F}\mathbf{H}_{1R}$ at U_1 and $\mathbf{H}_{R2}\mathbf{F}\mathbf{H}_{2R}$ at U_2, each of the two users can cancel out its self-interference. After removing the self-interferences, the signals at U_1 and U_2 are

$$\mathbf{y}'_i = \mathbf{H}_{Ri}\mathbf{F}\mathbf{H}_{\bar{i}R}\mathbf{x}_{\bar{i}} + \mathbf{H}_{Ri}\mathbf{F}\mathbf{n}_R + \mathbf{n}_i \tag{2.4}$$

It is useful to note that the two phases required here may correspond to two frequency channels or two time slots. If two time slots are used, the symbol vector $\mathbf{y}_R$ needs to be stored at R for at least one symbol interval. If two frequency channels are used, $\mathbf{y}_R$ needs

not to be stored at R, and the radio frequency output of R can be simply a frequency translation of its radio frequency input with the transformation $\mathbf{F}$. The implementation of $\mathbf{F}$ can be done digitally although no packet decoding or encoding is needed here. Both time division and frequency division for the two phases appear feasible.

Given the above two-way relay scheme and the signal model (2.4), the achievable data rate received at U_i is known to be $r_i = \frac{1}{2}\log_2 \det \mathbf{X}_i$ (under ideal encoding and decoding) where

$$\mathbf{X}_i = \mathbf{I} + (\mathbf{H}_{Ri}\mathbf{F}\mathbf{F}^H\mathbf{H}_{Ri}^H + \mathbf{I})^{-1}\mathbf{H}_{Ri}\mathbf{F}\mathbf{H}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{H}_{\bar{i}R}^H\mathbf{F}^H\mathbf{H}_{Ri}^H \tag{2.5}$$

The achievable (weighted) sum rate of this relay system is $r_{sum}(\mathbf{F}, \mathbf{R}_{x_1}, \mathbf{R}_{x_2}) = \mu_1 r_1 + \mu_2 r_2$, where the (non-negative) weights μ_1 and μ_2 can be chosen in any way subject to $\mu_1 + \mu_2 = 2$. For uniform weighting, we have $\mu_1 = \mu_2 = 1$.

The sum rate is a function of the source covariance matrices $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$ and the relay matrix $\mathbf{F}$. One important goal is to develop fast algorithms to determine these matrices to maximize the sum rate subject to power constraints at U_1, U_2 and R. This problem can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{F},\mathbf{R}_{x_1},\mathbf{R}_{x_2}} \quad & -r_{sum}(\mathbf{F}, \mathbf{R}_{x_1}, \mathbf{R}_{x_2}) \\ s.t. \quad & p_R(\mathbf{F}, \mathbf{R}_{x_1}, \mathbf{R}_{x_2}) \leq P_R, \mathrm{Tr}(\mathbf{R}_{x_i}) \leq P_i, i = 1, 2 \end{aligned} \tag{2.6}$$

This problem was not solved in the previous works [65], [37] and [77] except for the special case $N = 1$. When $N = 1$, there is no issue about the source matrices as $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$

reduce to scalars. With $N > 1$ and $M > 1$, this problem is a much harder (non-convex) problem and there is currently no globally optimal solution.

In this dissertation, we propose to optimize the pair $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$ and the single matrix $\mathbf{F}$ alternately. If we fix $\mathbf{R}_{x_1} = \mathbf{R}_{x_1}^{(k-1)}$ and $\mathbf{R}_{x_2} = \mathbf{R}_{x_2}^{(k-1)}$ where k be the index of the kth iteration, the problem of finding $\mathbf{F}$ is

$$\begin{aligned} \min_{\mathbf{F}} \quad & -r_{sum}(\mathbf{F}, \mathbf{R}_{x_1}^{(k-1)}, \mathbf{R}_{x_2}^{(k-1)}) \\ s.t. \quad & p_R(\mathbf{F}, \mathbf{R}_{x_1}^{(k-1)}, \mathbf{R}_{x_2}^{(k-1)}) \leq P_R \end{aligned} \tag{2.7}$$

where P_R is the upper bound on the transmit power at the relay. We will refer to this sub-problem as the relay optimization problem.

If we fix $\mathbf{F} = \mathbf{F}^{(k)}$, the problem of finding $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$ is

$$\begin{aligned} \min_{\mathbf{R}_{x_1}, \mathbf{R}_{x_2}} \quad & -r_{sum}(\mathbf{F}^{(k)}, \mathbf{R}_{x_1}, \mathbf{R}_{x_2}) \\ s.t. \quad & p_R(\mathbf{F}^{(k)}, \mathbf{R}_{x_1}, \mathbf{R}_{x_2}) \leq P_R \\ & \mathrm{Tr}(\mathbf{R}_{x_i}) \leq P_i, \ \mathbf{R}_{x_i} \geq 0, \ i = 1, 2 \end{aligned} \tag{2.8}$$

where P_i is the upper bound on the transmit power at U_i, and $\mathbf{R}_{x_i} \geq 0$ means that the matrix $\mathbf{R}_{x_i}$ is positive semi-definite. We will refer to this sub-problem as the source optimization problem.

Our approach to finding $\mathbf{R}_{x_1}$, $\mathbf{R}_{x_2}$ and $\mathbf{F}$ is based on the alternation of the optimizations of the above two sub-problems until convergence. At the time of this writing, we

do not know whether there exists a more effective approach. This alternate optimization between relay and sources is similar to one previously used for a one-way MIMO relay system of two or more hops [17], [49], [48]. But the problem here is much more complex and, to our knowledge, does not have the diagonalization property as available for the one-way MIMO relays.

Unlike the one-way (two-hop) scheme, the relay optimization problem here is still non-convex, for which we will present two algorithms in Section 2.4. The source optimization is convex, for which we will present an algorithm in Section 2.6 based on the generalized water-filling algorithm developed in [76]. In the next section, we show an optimal structure of the relay matrix, which is useful to reduce the complexity of the problem.

2.3 Optimal Structure of the Relay Matrix

Theorem 2.3.1. *Assume $M \geq 2N$. Define the following two QR decompositions:*

$$\begin{bmatrix} \mathbf{H}_{R1}^H & \mathbf{H}_{R2}^H \end{bmatrix} = \mathbf{V}_1 \mathbf{R}_1 \tag{2.9}$$

$$\begin{bmatrix} \mathbf{H}_{1R} & \mathbf{H}_{2R} \end{bmatrix} = \mathbf{U}_2 \mathbf{R}_2 \tag{2.10}$$

where $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{C}^{2N \times 2N}$ are upper triangular matrices, and $\mathbf{V}_1, \mathbf{U}_2 \in \mathbb{C}^{M \times 2N}$ are orthonormal matrices. Then, the optimal relay matrix $\mathbf{F}$ that maximizes the sum rate in problem (2.7) *has the following structure:*

$$\mathbf{F} = \mathbf{V}_1 \mathbf{A} \mathbf{U}_2^H \tag{2.11}$$

where $\mathbf{A} \in \mathbb{C}^{2N \times 2N}$.

Proof. See Appendix A.1. □

Note that this optimal structure of $\mathbf{F}$ is governed by the channel matrices only, which is not affected by the source covariance matrices. It is also clear from the proof that this structure of $\mathbf{F}$ remains optimal for any weights μ_1 and μ_2.

If $M > 2N$, then the M^2 unknowns in $\mathbf{F}$ are effectively reduced to $4N^2$ unknowns in $\mathbf{A}$ for the relay optimization problem (2.7). If $M = 2N$, this theorem does not seem important. With this theorem, we can now write

$$\mathbf{F} = \mathbf{S}\mathbf{A}\mathbf{T}^H \text{ where } \begin{cases} \mathbf{S} = \mathbf{V}_1, \mathbf{T} = \mathbf{U}_2 & \text{if } M > 2N \\ \mathbf{S} = \mathbf{T} = \mathbf{I} & \text{if } M \leq 2N \end{cases} \tag{2.12}$$

We also define $\mathbf{G}_{Ri} = \mathbf{H}_{Ri}\mathbf{S}$ and $\mathbf{G}_{iR} = \mathbf{T}^H\mathbf{H}_{iR}$. Then, (2.5) becomes

$$\mathbf{X}_i = \mathbf{I} + (\mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{I})^{-1}\mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H \tag{2.13}$$

and (2.2) becomes

$$p_R = \mathrm{Tr}\left(\mathbf{A}\mathbf{G}_{1R}\mathbf{R}_{x_1}\mathbf{G}_{1R}^H\mathbf{A}^H + \mathbf{A}\mathbf{G}_{2R}\mathbf{R}_{x_2}\mathbf{G}_{2R}^H\mathbf{A}^H + \mathbf{A}\mathbf{A}^H\right) \tag{2.14}$$

Although this theorem reduces the complexity (dimension of search space) of the problem when $M > 2N$, it does not change the structure of the remaining problem for finding $\mathbf{A}$. In other words, the problem (2.7) with respect to $\mathbf{F}$ has the same structure as the equivalent

problem with respect to $\mathbf{A}$. For convenience, we will treat $\mathbf{F}$ and $\mathbf{A}$ in (2.7) interchangeable.

It remains an open problem to find additional structure in the optimal $\mathbf{A}$. This difficulty is caused by the fact that the matrix $\mathbf{A}$ is weighted by different matrices in $\mathbf{X}_1$ and $\mathbf{X}_2$. In the next two sections, we present two algorithms for finding the optimal $\mathbf{A}$.

2.4 Relay Optimization by Gradient Method

In this section, we will present a hybrid gradient method that dynamically switches between steepest gradient descent and Newton's search. We need to solve the relay optimization problem (2.7), i.e., $\min_{\mathbf{A}} -r_{sum}$ subject to $p_R \leq P_R$. This is a non-convex problem because $-r_{sum}$ is not a convex function of $\mathbf{A}$. However, the set defined by $p_R \leq P_R$ with respect to $\mathbf{A}$ is a convex set, i.e., if $\mathbf{A}_1$ and $\mathbf{A}_2$ satisfy $p_R \leq P_R$, so does $\alpha\mathbf{A}_1 + (1-\alpha)\mathbf{A}_2$ where $0 \leq \alpha \leq 1$. Therefore, we can apply the interior point method to convert the hard constraint $p_R \leq P_R$ into a soft constraint in the following problem:

$$\min_{\mathbf{A}} f(\mathbf{A}) = -r_{sum} - \frac{1}{t}\ln(P_R - p_R) \tag{2.15}$$

where the second term in $f(\mathbf{A})$ is the soft constraint known as logarithmic barrier function [9], and t is the barrier factor. By increasing t gradually, the soft constraint becomes harder and harder. For each given t and a previous choice of $\mathbf{A}$ in the interior region (satisfying $p_R < P_R$), we can apply a gradient method to optimize $\mathbf{A}$. The loop of increasing t is called the outer loop, the gradient search under each fixed t is called the inner loop. There is no guarantee that this algorithm leads to the globally optimal solution unless the problem is

convex. Since $-r_{sum}$ is non-convex of $\mathbf{A}$, multiple initializations and multiple runs of the algorithm are desirable to increase the likelihood of finding the globally optimal solution.

The most common gradient methods are the (steepest) gradient descent method and the Newton's method. Both of these methods are well documented in textbooks on optimization, such as Algorithm 9.3 (gradient descent method) and Algorithm 9.5 (Newton's method) in [9]. The key component in the gradient descent method is the computation of the gradient vector ∇f of $f(\mathbf{A})$. In the Newton's method, we need the gradient vector ∇f as well as the Hessian matrix $\nabla^2 f$ of $f(\mathbf{A})$. The vector of independent variables here is $\mathbf{x} = [Re(\mathbf{a})^T, Im(\mathbf{a})^T]^T$ where $\mathbf{a} = \text{vec}(\mathbf{A})$. Hence, $\nabla f = \frac{\partial f}{\partial \mathbf{x}}$ and $\nabla^2 f = \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}^T}$. Alternatively, we can write $\nabla f = \left[\text{vec}^T\left(\frac{\partial f}{\partial Re(\mathbf{A})}\right), \text{vec}^T\left(\frac{\partial f}{\partial Im(\mathbf{A})}\right)\right]^T$ and

$$\nabla^2 f = \nabla(\nabla f)^T = \begin{bmatrix} \dfrac{\partial(\nabla f)^T}{\partial \text{vec}(Re(\mathbf{A}))} \\ \dfrac{\partial(\nabla f)^T}{\partial \text{vec}(Im(\mathbf{A}))} \end{bmatrix}$$

Next, we explain the key steps needed to derive and compute ∇f and $\nabla^2 f$.

2.4.1 Computation of Gradient

It follows from (2.15) that

$$\partial f(\mathbf{A}) = -\mu_1 \partial r_1 - \mu_2 \partial r_2 + \frac{1}{t(P_r - p_R)} \partial p_R \tag{2.16}$$

where $\partial r_i = \frac{1}{2}(\log_2 e)\text{Tr}(\mathbf{X}_i^{-1} \partial \mathbf{X}_i)$. For basic facts of matrix differential calculus, see [39]. Using (2.13), the chain rule of differentials, $\partial A^{-1} = -A^{-1} \partial A A^{-1}$ and $\text{Tr}(AB) = \text{Tr}(BA)$,

one can verify

$$\begin{aligned}\mathrm{Tr}(\mathbf{X}_i^{-1}\partial\mathbf{X}_i) = &-\mathrm{Tr}\left(\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\partial\mathbf{A}\right)\\ &-\mathrm{Tr}\left(\mathbf{G}_{Ri}^H\mathbf{D}^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_i^{-1}\mathbf{D}_1^{-1}\mathbf{G}_{Ri}\mathbf{A}\partial\mathbf{A}^H\right)\\ &+\mathrm{Tr}\left(\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\partial\mathbf{A}\right)\\ &+\mathrm{Tr}\left(\mathbf{G}_{Ri}^H\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\partial\mathbf{A}^H\right)\end{aligned} \tag{2.17}$$

where $\mathbf{D}_i = \mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{I}$ and $\mathbf{N}_{\bar{i}} = \mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\ \mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H$. With these notations, we also have $\mathbf{X}_i = \mathbf{I} + \mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}$. Using the fact $A(I+BA)^{-1} = (I+AB)^{-1}A$ (which follows from the matrix inversion lemma), one can verify that $(\mathbf{N}_{\bar{i}}\mathbf{X}_i^{-1})^H = \mathbf{N}_{\bar{i}}\mathbf{X}_i^{-1}$. One can also verify that if $\partial f = \mathrm{Tr}(A\partial X) + \mathrm{Tr}(A^H\partial X^H)$ then $\frac{\partial f}{\partial Re(X)} = 2Re(A)^T$ and $\frac{\partial f}{\partial Im(X)} = -2Im(A)^T$. We call the two terms in $\mathrm{Tr}(A\partial X) + \mathrm{Tr}(A^H\partial X^H)$ a conjugate differential pair. The first two terms in (2.17) are a conjugate differential pair, and so are the last two terms in (2.17). Therefore,

$$\begin{aligned}\frac{\partial r_i}{\partial Re(\mathbf{A})} = &-(\log_2 e)Re\left(\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T\\ &+(\log_2 e)Re\left(\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T\end{aligned} \tag{2.18}$$

and $\frac{\partial r_i}{\partial Im(\mathbf{A})}$ is the same as the above except that Re is replaced by $-Im$. Clearly,

$$\nabla r_i = \left[\mathrm{vec}^T\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right), \mathrm{vec}^T\left(\frac{\partial r_i}{\partial Im(\mathbf{A})}\right)^T\right]^T.$$

Using the same technique, we can find

$$\frac{\partial p_R}{\partial Re(\mathbf{A})} = 2Re(\mathbf{G}_{1R}\mathbf{R}_{x_1}\mathbf{G}_{1R}^H\mathbf{A}^H + \mathbf{G}_{2R}\mathbf{R}_{x_2}\mathbf{G}_{2R}^H\mathbf{A}^H + \mathbf{A}^H)^T$$

and $\frac{\partial p_R}{\partial Im(\mathbf{A})}$ is the same but with Re replaced by $-Im$. Using the above results, the gradient $\nabla f = -\mu_1\nabla r_1 - \mu_2\nabla r_2 + \frac{1}{t(P_r - p_R)}\nabla p_R$ can be determined.

2.4.2 Computation of Hessian

To compute the Hessian $\nabla^2 f$, we need to compute each term in $\nabla^2 f = -\mu_1\nabla^2 r_1 - \mu_2\nabla^2 r_2 + \frac{1}{t(P_r - p_R)}\nabla^2 p_R$. The procedure of finding $\nabla^2 p_R$ is similar to that of finding $\nabla^2 r_i$ for $i = 1, 2$. To find $\nabla^2 r_i = \nabla(\nabla r_i)^T$, we need to find the elements $\frac{\partial\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$, $\frac{\partial\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right)_{m,n}}{\partial Im(\mathbf{A})_{k,l}}$, $\frac{\partial\left(\frac{\partial r_i}{\partial Im(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$ and $\frac{\partial\left(\frac{\partial r_i}{\partial Im(\mathbf{A})}\right)_{m,n}}{\partial Im(\mathbf{A})_{k,l}}$ where $1 \le m, n, k, l \le M_0 \doteq \min(2N, M)$. In other words, each entry of the $2M_0^2 \times 2M_0^2$ matrix $\nabla^2 r_i$ is determined by one of the four elements for some m, n, k, l.

To find $\frac{\partial\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$, we need to take the first order differential of (2.18) to obtain

$$\begin{aligned}
\partial\frac{\partial r_i}{\partial Re(\mathbf{A})} =& -(\log_2 e)Re\left(\partial\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_{\bar{i}}^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&+(\log_2 e)Re\left(\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\partial\mathbf{D}_i\mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_{\bar{i}}^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&-(\log_2 e)Re\left(\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\partial\mathbf{N}_{\bar{i}}\mathbf{X}_{\bar{i}}^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&+(\log_2 e)Re\left(\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_{\bar{i}}^{-1}\partial\mathbf{X}_i\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&+(\log_2 e)Re\left(\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{D}_i^{-1}\mathbf{N}_{\bar{i}}\mathbf{X}_{\bar{i}}^{-1}\mathbf{D}_i^{-1}\partial\mathbf{D}_i\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&+(\log_2 e)Re\left(\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\partial\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&-(\log_2 e)Re\left(\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{X}_i^{-1}\partial\mathbf{X}_i\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \\
&-(\log_2 e)Re\left(\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{X}_i^{-1}\mathbf{D}_i^{-1}\partial\mathbf{D}_i\mathbf{D}_i^{-1}\mathbf{G}_{Ri}\right)^T \qquad (2.19)
\end{aligned}$$

where $\partial\mathbf{D}_i$, $\partial\mathbf{N}_{\bar{i}}$ and $\partial\mathbf{X}_i$ can be determined similarly such that the resulting expression is a linear function of $\partial\mathbf{A}$ and $\partial\mathbf{A}^H$. To obtain $\frac{\partial\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$ from $\partial\frac{\partial r_i}{\partial Re(\mathbf{A})}$, we simply choose the (m,n)th element of $\partial\frac{\partial r_i}{\partial Re(\mathbf{A})}$, replace $\partial\mathbf{A}$ by $\mathbf{e}_k\mathbf{e}_l^T$ and replace $\partial\mathbf{A}^H$ by $\mathbf{e}_l\mathbf{e}_k^T$, where all entries of the vector $\mathbf{e}_k$ are zero except that its kth entry is one.

The expression of $\partial\frac{\partial r_i}{\partial Im(\mathbf{A})}$ is also given by (2.19) except that the operator Re should be replaced by $-Im$. Then, finding $\frac{\partial\left(\frac{\partial r_i}{\partial Im(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$ from $\partial\frac{\partial r_i}{\partial Im(\mathbf{A})}$ follows the same procedure as above.

The derivation for $\frac{\partial\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right)_{m,n}}{\partial Im(\mathbf{A})_{k,l}}$ and $\frac{\partial\left(\frac{\partial r_i}{\partial Im(\mathbf{A})}\right)_{m,n}}{\partial Im(\mathbf{A})_{k,l}}$ is the same as for $\frac{\partial\left(\frac{\partial r_i}{\partial Re(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$ and $\frac{\partial\left(\frac{\partial r_i}{\partial Im(\mathbf{A})}\right)_{m,n}}{\partial Re(\mathbf{A})_{k,l}}$, respectively, except that for the former pair we should replace $\partial\mathbf{A}$ by $j\mathbf{e}_k\mathbf{e}_l^T$ and $\partial\mathbf{A}^H$ by $-j\mathbf{e}_l\mathbf{e}_k^T$.

The details of those expressions are tedious and omitted to save space. However, for computational efficiency, it is important to program the computations by starting from the sparse vector $\mathbf{e}_k$.

2.4.3 Hybrid Gradient Method

Since the function $f(\mathbf{A})$ is non-convex, there is no guarantee that the Hessian $\nabla^2 f$ is positive definite everywhere while a positive definite $\nabla^2 f$ is a necessary condition for the Newton's method to converge to even a local minimum. In fact, our simulation results show that the Newton's method does not always converge to a local minimum, and as a consequence the gradient descent method can sometimes yield a better result. Therefore, we propose a hybrid gradient method that combine the two gradient methods. The procedure of the hybrid gradient method is simple. At each iteration of the inner loop (under a fixed t), we compute both the gradient ∇f and the Hessian $\nabla^2 f$. If $\nabla^2 f$ is singular or the Newton's decrement $\lambda^2 = \nabla f \left(\nabla^2 f\right)^{-1} \nabla f$ is less than or equal to zero, we follow the gradient descent method. Otherwise, we choose the search (either Newton's or gradient descent) that produces a larger descent of $f(\mathbf{A})$. The iterations of the Newton's method stop when λ^2 is small enough. When $\mathbf{A}$ is close enough to a local minimum of $f(\mathbf{A})$, $\nabla^2 f$ is typically positive definite and in this case the Newton's method is known to have a quadratic convergence rate. For the gradient descent method, the convergence rate is known to be linear. In simulation, the hybrid gradient method has consistently produced either the same or better results than the pure Newton's method and the pure gradient descent method.

The hybrid gradient method is summarized as follows:

Algorithm 2.1 (Hybrid gradient method)**.**

1: ***Initialization:*** *Choose* $\epsilon > 0$, $\eta > 1$, $\alpha \in (0.01, 0.3)$, $\beta \in (0.1, 0.8)$. *Initialize a feasible* $\mathbf{A}$ *or its equivalent* $\mathbf{x}$. *Set* $t = t_0$. *Note that the choice of* ϵ *governs the precision of the final result. See Step 6). The choice of* η *(as long as larger than one) is not critical. See page 570 of [9]. The choices of* $\alpha \in (0.01, 0.3)$ *and* $\beta \in (0.1, 0.8)$ *are empirical as recommended on page 466 of [9].*

2: ***repeat***

3: ***repeat***

4: *Compute* ∇f *and* $\nabla^2 f$.

5: *Set* $t^{(i)} = 1$ *for* $i = 1, s$.

6: *Line backtracking:*

7: ***for*** $i = 1, s$ ***do***

8: ***while*** $f(\mathbf{x} + t^{(i)}\Delta\mathbf{x}^{(i)}) > f(\mathbf{x}) + \alpha t^{(i)}(\nabla f)^T \Delta\mathbf{x}^{(i)}$ ***do***

9: *Set* $t^{(i)} := \beta t^{(i)}$.

10: ***end while***

11: ***end for***

12: *Update:*

13: ***if*** $f(\mathbf{x} + t^{(2)}\Delta\mathbf{x}^{(2)}) \leq f(\mathbf{x} + t^{(1)}\Delta\mathbf{x}^{(1)})$ ***then***

14: *Let* $\mathbf{x} := \mathbf{x} + t^{(2)}\Delta\mathbf{x}^{(2)}$.

15: ***else***

16: *Let* $\mathbf{x} := \mathbf{x} + t^{(1)}\Delta\mathbf{x}^{(1)}$.

17: ***end if***

18: ***until*** $-(\nabla f)^T \Delta \mathbf{x} < \epsilon$

19: *Set* $t := \eta t$.

20: ***until*** $1/t < \epsilon$

For convenience, we have used the same tolerance for both Steps 6 and 7 although different choices can be made. The inverse of $\nabla^2 f$ needs not and should not to be computed explicitly in order to compute $(\nabla^2 f)^{-1}\nabla f$ efficiently. It can be found by solving efficiently the linear equation $(\nabla^2 f)\mathbf{y} = \nabla f$. In simulation, we will choose $\epsilon = 10^{-3}$, $\eta = 3$, $\alpha = 0.01$, $\beta = 0.5$ and $t_0 = 1$. Only a locally optimal solution can be found by the hybrid gradient method with a given initial choice of $\mathbf{A}$. To increase the likelihood of finding the globally optimal solution, multiple initializations and multiple runs of the search algorithm are useful.

2.5 Relay Optimization by Iterative WMMSE Method

In this section, we present an alternative algorithm for the relay optimization problem (2.7) with respect to $\mathbf{A}$. Recall $r_{sum} = \frac{1}{2}\sum_{i=1}^{2} \mu_i \log_2 \det(\mathbf{X}_i)$ where $\mathbf{X}_i$ is given in (2.13). Since $\det(I + AB) = \det(I + BA)$, we can write $r_{sum} = \frac{1}{2}\sum_{i=1}^{2} \mu_i \log_2 \det(\mathbf{Y}_i)$ where

$$\mathbf{Y}_i = \mathbf{I} + \mathbf{R}_{x_{\bar{i}}}^{H/2} \mathbf{G}_{\bar{i}R}^H \mathbf{A}^H \mathbf{G}_{Ri}^H \left(\mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{I}\right)^{-1} \mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}^{1/2} \tag{2.20}$$

We define $J_i = \mathrm{Tr}\left(\mathbf{W}_i\mathbf{E}_i\right) - \log\det(\mathbf{W}_i) - N$ where $\mathbf{E}_i = E\left(\left(\mathbf{Q}_i\mathbf{y}_i' - \mathbf{s}_{\bar{i}}\right)\left(\mathbf{Q}_i\mathbf{y}_i' - \mathbf{s}_{\bar{i}}\right)^H\right)$

and $\mathbf{y}_i'$ is given in (2.4) or equivalently

$$\mathbf{y}_i' = \mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}^{1/2}\mathbf{s}_{\bar{i}} + \mathbf{G}_{Ri}\mathbf{A}\mathbf{n}_R + \mathbf{n}_i \tag{2.21}$$

We next consider the following alternative problem

$$\begin{aligned} \min_{\mathbf{A},\mathbf{Q}_i,\mathbf{W}_i,i=1,2} \quad & J = \mu_1 J_1 + \mu_2 J_2 \\ s.t. \quad & p_R(\mathbf{A}) \le P_R \end{aligned} \tag{2.22}$$

It is easy to see that the optimal solution of $\mathbf{Q}_i$ from (2.22) is $\mathbf{Q}_i = E(\mathbf{s}_{\bar{i}}\mathbf{y}_i'^H)\cdot \left(E(\mathbf{y}_i'\mathbf{y}_i'^H)\right)^{-1}$. Using (2.21), one can verify that

$$\begin{aligned} \mathbf{Q}_i =& \mathbf{R}_{x_{\bar{i}}}^{\frac{H}{2}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\left(\mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{I}\right)^{-1} \\ =& \left(\mathbf{I} + \mathbf{R}_{x_{\bar{i}}}^{\frac{H}{2}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\left(\mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{I}\right)^{-1}\mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}^{\frac{1}{2}}\right)^{-1} \\ & \mathbf{R}_{x_{\bar{i}}}^{\frac{H}{2}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\left(\mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H + \mathbf{I}\right)^{-1} \end{aligned} \tag{2.23}$$

where the last equation follows from the matrix inverse lemma. With the definition of $\mathbf{Y}_i$ in (2.20) and the optimal $\mathbf{Q}_i$ in (2.23), it is easy to verify that $\mathbf{E}_i = \mathbf{Y}_i^{-1}$. It is also an easy task to verify that the optimal solution $\mathbf{W}_i$ from (2.22) is simply $\mathbf{W}_i = \mathbf{Y}_i$. We see that with the optimal $\mathbf{Q}_i$ and $\mathbf{W}_i$, $J_i = -\log\det(\mathbf{Y}_i)$ and hence $J = -2(\log 2)r_{sum}$. Therefore, we have shown that the optimal solution $\mathbf{A}$ to (2.22) is the same as that to the original problem (2.7).

Furthermore, it is important to notice that if we fix $\{\mathbf{Q}_i,\ i = 1, 2\}$ and $\{\mathbf{W}_i,\ i =$

$1, 2\}$, the problem of (2.22) with respect to $\mathbf{A}$ is a quadratic convex problem. Therefore, we can try to find the solution to (2.22) by optimizing $\{\mathbf{Q}_i,\ i = 1, 2\}$, $\{\mathbf{W}_i,\ i = 1, 2\}$ and $\mathbf{A}$ in a cyclic fashion, which is the following iterative WMMSE algorithm:

Algorithm 2.2 (iterative WMMSE algorithm)**.**

1: ***Given*** $\forall \mathbf{A}^{(0)}$*;*

2: ***repeat***

3: *Update* $k := k + 1$*;*

4: *Compute* $\mathbf{Q}_i^{(k)}$ *based on* $\mathbf{A}^{(k-1)}$ *(see* (2.23)*);*

5: *Compute* $\mathbf{W}_i^{(k)} = \mathbf{Y}_i^{(k)}$ *based on* $\mathbf{A}^{(k-1)}$ *(see* (2.20) *)*

6: *Compute* $\mathbf{A}^{(k)}$ *by solving* (2.22) *and fixing* $\mathbf{Q}_i = \mathbf{Q}_i^{(k)}$ *and* $\mathbf{W}_i = \mathbf{W}_i^{(k)}$.

7: ***until*** *Convergence*

Although there is no proof or disproof that the iterative WMMSE algorithm yields the optimal $\mathbf{A}$ of the original problem (2.7), this algorithm is guaranteed to converge locally due to the minimization of J with respect to $\{\mathbf{Q}_i,\ i = 1, 2\}$, $\{\mathbf{W}_i,\ i = 1, 2\}$ and $\mathbf{A}$ cyclically.

We next show how to implement Step 3 of the above algorithm. First, using (2.21) and performing the expectation in $\mathbf{E}_i$, one can verify that, after removing the constant term $\log\det(\mathbf{W}_i) + N$,

$$\begin{aligned} J_i =& \mathrm{Tr}(\mathbf{W}_i\mathbf{Q}_i\mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{Q}_i^H) \\ &+ \mathrm{Tr}(\mathbf{W}_i\mathbf{Q}_i\mathbf{G}_{Ri}\mathbf{A}\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{Q}_i^H) + \mathrm{Tr}(\mathbf{W}_i\mathbf{Q}_i\mathbf{Q}_i^H) \\ &+ \mathrm{Tr}(\mathbf{W}_i) - \mathrm{Tr}(\mathbf{W}_i\mathbf{Q}_i\mathbf{G}_{Ri}\mathbf{A}\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}^{\frac{1}{2}}) - \mathrm{Tr}(\mathbf{W}_i\mathbf{R}_{x_{\bar{i}}}^{\frac{H}{2}}\mathbf{G}_{\bar{i}R}^H\mathbf{A}^H\mathbf{G}_{Ri}^H\mathbf{Q}_i^H) \end{aligned} \tag{2.24}$$

Recall $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$, $\mathrm{Tr}(AB^H) = \mathrm{vec}^H(B)\mathrm{vec}(A)$, $\mathrm{Tr}(ABA^HC^H) = \mathrm{vec}^H(CA)\mathrm{vec}(AB)$ $= \mathrm{vec}(A)^H(I \bigotimes C)^H\ (B^T \bigotimes I)\mathrm{vec}(A) = \mathrm{vec}^H(A)(B^T \bigotimes C^H)\mathrm{vec}(A)$. It follows that

$$J_i = \mathbf{a}^H\mathbf{A}_i\mathbf{a} - \mathbf{c}_i^H\mathbf{a} - \mathbf{a}^H\mathbf{c}_i + d_i \tag{2.25}$$

where $\mathbf{a} = \mathrm{vec}(\mathbf{A})$ and

$$\mathbf{A}_i \quad = \quad (\mathbf{G}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{G}_{\bar{i}R}^H)^T \otimes (\mathbf{G}_{Ri}^H\mathbf{Q}_i^H\mathbf{W}_i\mathbf{Q}_i\mathbf{G}_{Ri}) + \mathbf{I} \otimes (\mathbf{G}_{Ri}^H\mathbf{Q}_i^H\mathbf{W}_i\mathbf{Q}_i\mathbf{G}_{Ri}) \tag{2.26}$$

$$\mathbf{c}_i = \mathrm{vec}(\mathbf{G}_{Ri}^H\mathbf{Q}_i^H\mathbf{W}_i\mathbf{R}_{x_{\bar{i}}}^{\frac{H}{2}}\mathbf{G}_{\bar{i}R}^H) \tag{2.27}$$

$$d_i = \mathrm{Tr}(\mathbf{W}_i\mathbf{Q}_i\mathbf{Q}_i^H) + \mathrm{Tr}(\mathbf{W}_i) \tag{2.28}$$

Also, we can write (2.14) as $p_R = \mathbf{a}^H\mathbf{G}_R\mathbf{a}$ with $\mathbf{G}_R = (\mathbf{G}_{1R}\mathbf{R}_{x_1}\mathbf{G}_{1R}^H + \mathbf{G}_{2R}\mathbf{R}_{x_2}\mathbf{G}_{2R}^H + \mathbf{I})^T \bigotimes \mathbf{I}$

Therefore, Step 3 is equivalent to

$$\begin{aligned} \min_{\mathbf{a}} \quad & \mathbf{a}^H\mathbf{G}_0\mathbf{a} - \mathbf{c}^H\mathbf{a} - \mathbf{a}^H\mathbf{c} \\ s.t. \quad & \mathbf{a}^H\mathbf{G}_R\mathbf{a} - P_R \leq 0 \end{aligned} \tag{2.29}$$

where $\mathbf{G}_0 = \mu_1\mathbf{A}_1 + \mu_2\mathbf{A}_2$ and $\mathbf{c} = \mu_1\mathbf{c}_1 + \mu_2\mathbf{c}_2$. This is a simple convex problem which can be solved by the KKT method [9]. Specifically, the optimal $\mathbf{a}$ is uniquely determined

by the following equations:

$$\begin{cases} \mathbf{G}_0\mathbf{a} - \mathbf{c} + \xi\mathbf{G}_R\mathbf{a} = \mathbf{0} \\ \xi(\mathbf{a}^H\mathbf{G}_R\mathbf{a} - P_R) = 0 \\ \mathbf{a}^H\mathbf{G}_R\mathbf{a} - P_R \leq 0 \end{cases} \tag{2.30}$$

where $\xi \in \mathbb{R}$ and $\xi \geq 0$. There are two possible cases for the optimal solution $\mathbf{a}$. The first is when the constraint is not active, i.e., $\mathbf{a}^H\mathbf{G}_R\mathbf{a} < P_R$ or equivalently $\xi = 0$, and the second is when the constraint is active, i.e., $\mathbf{a}^H\mathbf{G}_R\mathbf{a} = P_R$ or equivalently $\xi > 0$. In the first case, the solution is $\mathbf{a} = \mathbf{G}_0^{-1}\mathbf{c}$. If this solution does not meet the power constraint, we can simply abandon it and consider the second case. In the second case, we have $\mathbf{a} = (\mathbf{G}_0 + \xi\mathbf{G}_R)^{-1}\mathbf{c}$ where $\xi > 0$ is such that

$$h(\xi) = \mathbf{c}^H(\mathbf{G}_0 + \xi\mathbf{G}_R)^{-1}\mathbf{G}_R(\mathbf{G}_0 + \xi\mathbf{G}_R)^{-1}\mathbf{c} = P_R \tag{2.31}$$

which can be solved by a 1-D search such as the bisection method since $h(\xi)$ is monotonically decreasing function of ξ. In simulation, we will choose the error tolerance $\epsilon = 10^{-3}$ for the 1-D search.

The iterative WMMSE method was inspired by the work [11] where a similar idea was used for MIMO broadcast. In Section 2.7, we will compare the iterative WMMSE method with the hybrid gradient method.

2.6 Source Optimization by Generalized Water Filling

We now consider the source optimization problem (2.8) with any fixed $\mathbf{F}$. To highlight $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$ which are the variables of interest now, we first reformulate (2.8) as (ignoring the factor 1/2):

$$\begin{aligned} \min_{\mathbf{R}_{x_1},\mathbf{R}_{x_1}} \quad & -\mu_2 \log_2 \det(\mathbf{I} + \mathbf{H}_1 \mathbf{R}_{x_1} \mathbf{H}_1^H) \\ & -\mu_1 \log_2 \det(\mathbf{I} + \mathbf{H}_2 \mathbf{R}_{x_2} \mathbf{H}_2^H) \\ s.t. \quad & \mathrm{Tr}(\mathbf{R}_{x_i}) \le P_i, \quad \mathbf{R}_{x_i} \ge 0, \quad i = 1, 2 \\ & \mathrm{Tr}(\mathbf{G}_1 \mathbf{R}_{x_1} \mathbf{G}_1^H + \mathbf{G}_2 \mathbf{R}_{x_2} \mathbf{G}_2^H) \le P_3 \end{aligned} \tag{2.32}$$

where $\mathbf{H}_i = \left(\mathbf{H}_{iR}^H \mathbf{F}^H \mathbf{H}_{R\bar{i}}^H (\mathbf{H}_{R\bar{i}} \mathbf{F}\mathbf{F}^H \mathbf{H}_{R\bar{i}}^H + \mathbf{I})^{-1} \mathbf{H}_{R\bar{i}} \mathbf{F}\ \mathbf{H}_{iR}\right)^{\frac{1}{2}}$, $\mathbf{G}_i = \mathbf{F}\mathbf{H}_{iR}$, $P_3 = P_R - \mathbf{F}\mathbf{F}^H$. For any μ_1 and μ_2, the above problem is convex and can be solved by (general purpose) semi-definite programming (SDP) such as in CVX [19] . But if $\mu_1 = \mu_2$, there is an alternative method shown below.

With the uniform weights, we can write (2.32) as

$$\begin{aligned} \min_{\mathbf{R}_x \ge 0} \quad & -\log_2 \det(\mathbf{I} + \mathbf{H}\mathbf{R}_x\mathbf{H}^H) \\ s.t. \quad & \mathrm{Tr}(\mathbf{B}_i \mathbf{R}_x \mathbf{B}_i^H) \le P_i, \quad i = 1, 2, 3 \end{aligned} \tag{2.33}$$

where $\mathbf{H} = \mathrm{diag}(\mathbf{H}_1, \mathbf{H}_2)$, $\mathbf{B}_1 = \mathrm{diag}(\mathbf{I}, \mathbf{0})$, $\mathbf{B}_2 = \mathrm{diag}(\mathbf{0}, \mathbf{I})$, $\mathbf{B}_3 = \mathrm{diag}(\mathbf{G}_1, \mathbf{G}_2)$. It is clear that if the problem (2.33) has the diagonal constraint $\mathbf{R}_x = \mathrm{diag}(\mathbf{R}'_{x_1}, \mathbf{R}'_{x_2})$ where $\mathbf{R}'_{x_1}$ and $\mathbf{R}'_{x_2}$ have the same dimensions as $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$, then (2.32) and (2.33) are exactly the same

problem and hence have the same solution, i.e., $\mathbf{R}'_{x_1} = \mathbf{R}_{x_1}$ and $\mathbf{R}'_{x_2} = \mathbf{R}_{x_2}$.

Next, we show that $\mathbf{R}_x = \text{diag}(\mathbf{R}'_{x_1}, \mathbf{R}'_{x_2})$ is always the form of the solution to (2.33). Without loss of generality, we can write

$$\mathbf{R}_x = \begin{bmatrix} \mathbf{R}'_{x_1} & \mathbf{R} \\ \mathbf{R}^H & \mathbf{R}'_{x_2} \end{bmatrix}$$

It is easy to see that due to the structures of $\mathbf{H}$, $\mathbf{B}_1$, $\mathbf{B}_2$ and $\mathbf{B}_3$, the off-diagonal block $\mathbf{R}$ of $\mathbf{R}_x$ has no effect on the constraints in (2.33). Now, we focus on the objective function of (2.33), for which we know, due to the Fischer's inequality [25], that

$$\begin{aligned} \det(\mathbf{I} + \mathbf{H}\mathbf{R}_x\mathbf{H}^H) &= \det \begin{bmatrix} \mathbf{I} + \mathbf{H}_1\mathbf{R}'_{x_1}\mathbf{H}_1^H & \mathbf{H}_1\mathbf{R}\mathbf{H}_2^H \\ \mathbf{H}_2\mathbf{R}^H\mathbf{H}_1^H & \mathbf{I} + \mathbf{H}_2\mathbf{R}'_{x_2}\mathbf{H}_2^H \end{bmatrix} \\ &\leq \det \begin{bmatrix} \mathbf{I} + \mathbf{H}_1\mathbf{R}'_{x_1}\mathbf{H}_1^H & \mathbf{0} \\ \mathbf{0} & \mathbf{I} + \mathbf{H}_2\mathbf{R}'_{x_2}\mathbf{H}_2^H \end{bmatrix} \\ &= \det(\mathbf{I} + \mathbf{H}\text{diag}(\mathbf{R}'_{x_1}, \mathbf{R}'_{x_2})\mathbf{H}^H) \end{aligned} \tag{2.34}$$

Therefore, the solution to (2.33) must be block diagonal.

For (2.33), we can directly apply the generalized water filling (GWF) theorem from [76], which yields

$$\mathbf{R}_x = \mathbf{K}^{-H}\mathbf{V}(\mathbf{I} - \mathbf{\Sigma}^{-2})^+\mathbf{V}^H\mathbf{K}^{-1} \tag{2.35}$$

where $\mathbf{K} = (\sum_{i=1}^{3} \theta_i \mathbf{B}_i^H \mathbf{B}_i)^{\frac{1}{2}}$, $\mathbf{V}$ and $\mathbf{\Sigma}$ are given by the SVD $\mathbf{H}\mathbf{K}^{-H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$, and $x^+ = \max(x, 0)$ is applied to each diagonal element in $(\mathbf{I} - \mathbf{\Sigma}^{-2})^+$. The Lagrangian variables

$\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ are the solution to the dual problem

$$\begin{aligned} \max_{\boldsymbol{\theta} \geq \mathbf{0}} \quad & -\log_2 \det(\mathbf{I} + \mathbf{H}\mathbf{R}_x\mathbf{H}^H) + \sum_{i=1}^{3} \theta_i \big(\mathrm{Tr}(\mathbf{B}_i\mathbf{R}_x\mathbf{B}_i^H) - P_i'\big) \\ s.t. \quad & \mathbf{R}_x = \mathbf{K}^{-H}\mathbf{V}(\mathbf{I} - \boldsymbol{\Sigma}^{-2})^+\mathbf{V}^H\mathbf{K}^{-1} \end{aligned} \tag{2.36}$$

The dual problem is convex and can be solved by the Newton's method [76]. In our Matlab simulation, the GWF algorithm is much faster than CVX to solve (2.33).

Finally, we would like to add that the source optimization problem (2.32) differs from a conventional single-link MIMO channel problem such as in [60]. For the former, the optimal source matrices are not necessarily diagonal in general.

2.7 Simulation Results

For simulation, we choose the uniform weights $\mu_1 = \mu_2 = 1$, and the elements in $\mathbf{H}_{1R}$, $\mathbf{H}_{2R}$, $\mathbf{H}_{R1}$, and $\mathbf{H}_{R2}$ as independent complex circular Gaussian random variables of zero mean and unit variance, i.e., $\mathcal{CN}(0, 1)$. For illustration of the convergence behaviors of our algorithms, we will use one realization of the channel matrices. For illustration of the averaged sum rate, we will use multiple (100) realizations of the channel matrices. We assume that all noise vectors are complex white Gaussian, i.e., $\mathbf{n}_R \sim \mathcal{CN}(0, \mathbf{I})$ and $\mathbf{n}_i \sim \mathcal{CN}(0, \mathbf{I})$ where $i = 1, 2$. We denote $\mathrm{SNR}_R = P_R/M$ and $\mathrm{SNR}_i = P_i/N$ where $i = 1, 2$.

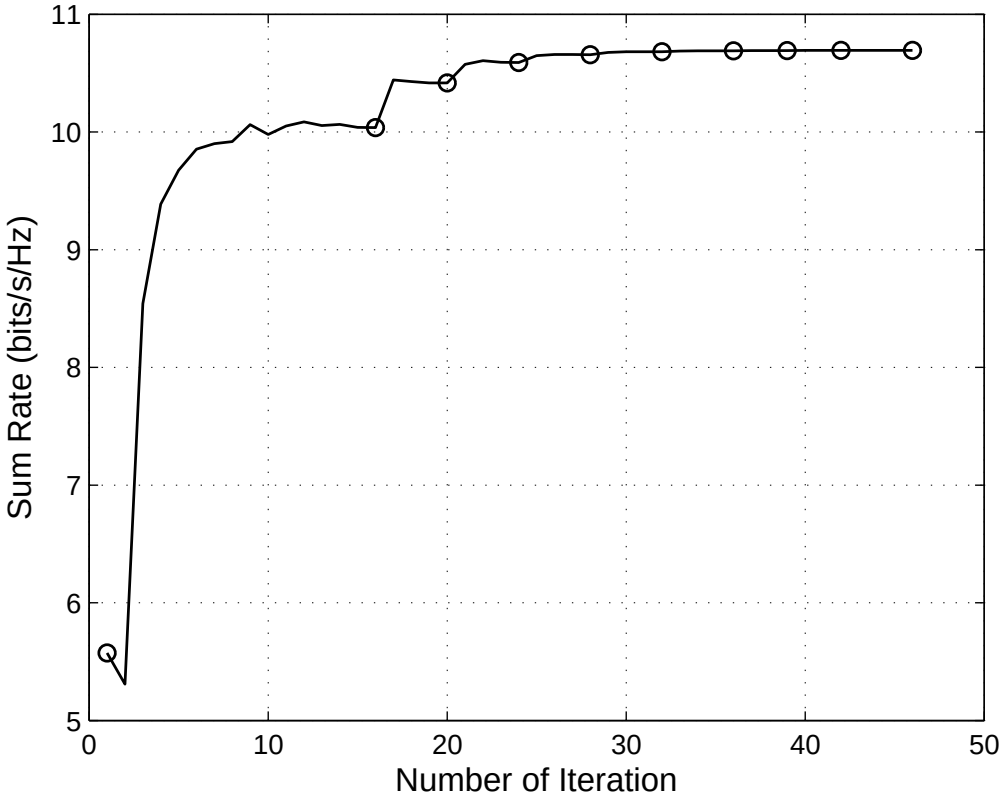

Figure 2.1: The sum rate by the hybrid gradient method versus its iteration. The circles indicate the iteration steps where t is increased by the factor three, and the iterations between two adjacent circles are the iterations of either the (steepest) gradient descent method or Newton's method for a fixed t. $N = 2$ and $M = 6$. $\mathrm{SNR}_1 = \mathrm{SNR}_2 = \mathrm{SNR}_R = 10dB$.

2.7.1 Relay Optimization

In this subsection, we assume that $\mathbf{R}_{\mathbf{x}_1}$ and $\mathbf{R}_{\mathbf{x}_2}$ are diagonal matrix with identical entries subject to $\mathrm{Tr}(\mathbf{R}_{\mathbf{x}_1}) = P_1$ and $\mathrm{Tr}(\mathbf{R}_{\mathbf{x}_2}) = P_2$. We randomly generate the initial $\mathbf{A}$ satisfying $p_R(\mathbf{A}) = P_R - \theta$ where θ is a (small) positive value so that $-\ln(P_R - p_R(\mathbf{A}))$ is not too large. This condition is required for the gradient methods although it is not necessary for the iterative MMSE method.

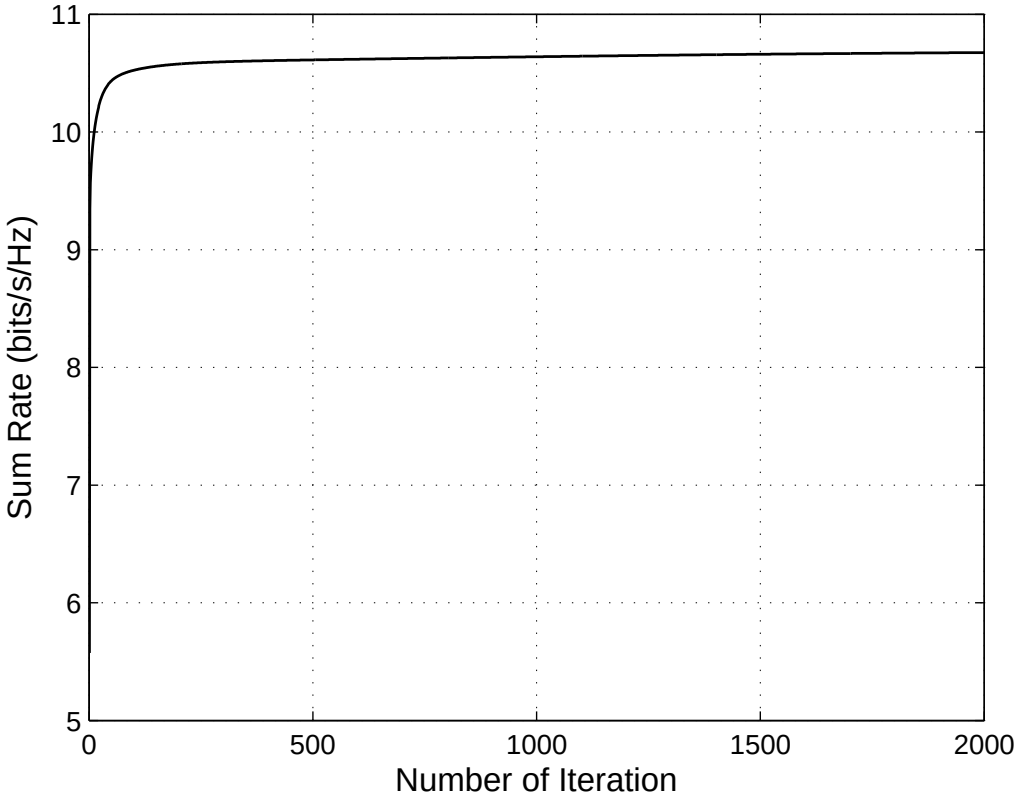

Figure 2.2: The sum rate by the iterative WMMSE method versus its iteration. $N = 2$ and $M = 6$. $\mathrm{SNR}_1 = \mathrm{SNR}_2 = \mathrm{SNR}_R = 10dB$.

Note that the hybrid gradient method is always much faster than the gradient descend method and yields the same or better result than the Newton's method. Only the comparison between the hybrid gradient method and the iterative WMMSE method is shown next.

Under all common initializations of $\mathbf{A}$, the hybrid gradient method and the iterative WMMSE method have consistently yielded the same maximized sum rate upon convergence. Fig. 2.1 illustrates the convergence behavior of the hybrid gradient method, which is the value of the sum rate versus the iteration index. The circles indicate the lo-

cations along the iteration process where the barrier constant $t = 3^n$ is increased by the factor three. Fig 2.2 illustrates the convergence behavior of the iterative WMMSE method versus its iteration cycle of renewing $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{W}_1$ and $\mathbf{W}_2$ in the cost $J_1 + J_2$.

To provide a more precise comparison of the speed of the two methods, Fig. 2.3 illustrates the sum rate by the hybrid gradient method and the iterative WMMSE method versus the actual computational time (in Matlab on the same computer) along the iteration process. We call such curves as time-capacity curves. (Undoubtedly, these curves would be affected by how each algorithm is implemented. But we believe that these curves are meaningful to reflect the major performance gaps between algorithms.) Each point on a time-capacity curve is a pair of values of time and sum rate. The time is the time an algorithm takes to produce the corresponding sum rate. For each of the two methods, we embedded stopwatch check points in the programs, and the times (in seconds) and the corresponding values of intermediate $\mathbf{A}$ are collected. For each $\mathbf{A}$, there is a corresponding value of the sum rate. We see that when the sum rate is close to the optimal, the hybrid gradient method is much faster. Note that the Newton's method has a quadratic convergence rate near the optimal point. On the other hand, when the sum rate is far from the optimal, the iterative WMMSE method converges faster initially.

We like to mention that the method shown in [62] is a non-iterative sub-optimal method. This method is faster than a single iteration of the hybrid gradient or the iterative WMMSE method. However, using the method in [62], we obtained the sum rate at about 8.5 bits/s/Hz, which is significantly smaller than the maximum sum rate shown in Fig. 2.3.

The work [77] considered a special case of the two-way MIMO relay system where

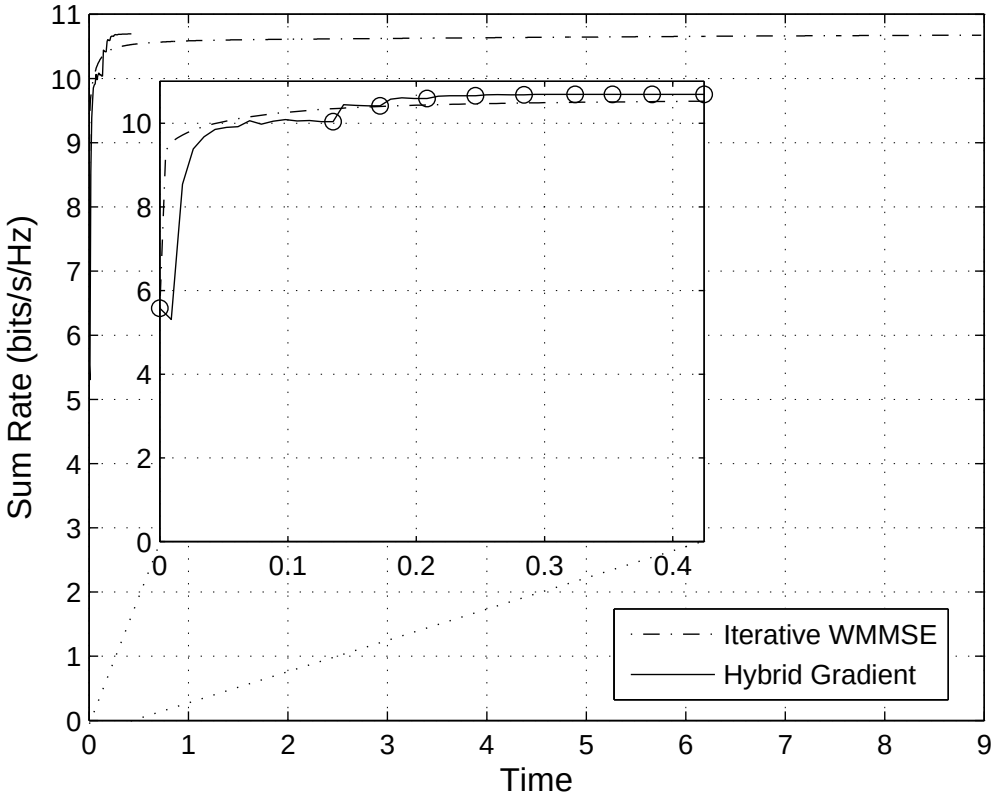

Figure 2.3: The time-capacity curves of the hybrid gradient method and the iterative WMMSE method. The time is in seconds. $N = 2$ and $M = 6$. $\mathrm{SNR}_1 = \mathrm{SNR}_2 = \mathrm{SNR}_R = 10dB$.

each user has a single antenna. In this case, there is no need for source optimization but only relay optimization. In [77], an algorithm based on SDP was developed to compute the capacity region of this special case. This algorithm can be easily adopted to compute the maximum sum rate as well. In Appendix A.2, a "zoomed" SDP algorithm is formulated to compute the maximum sum rate, which is a modified (and faster) version of the original SDP algorithm shown in [77]. Upon convergence, the zoomed SDP method has consistently produced the same sum rates as the hybrid gradient and iterative WMMSE methods under

the same conditions. Fig. 2.4 illustrates the time-capacity curves of the three methods for $N = 1$ and $M = 5$. In this figure, the zoomed SDP method is about 1000 times slower than the hybrid gradient method to yield the same maximized sum rate.

2.7.2 Joint Source-Relay Optimization

When $N > 1$ and $M > 1$, both the relay optimization and the source optimization are important. We now let $\mathrm{SNR}_1 = \mathrm{SNR}_2 = \mathrm{SNR}_R = \mathrm{SNR}$. Fig. 2.5 shows the averaged sum rates achieved by different schemes where SNR varies from 10dB to 50dB. We used 100 channel realizations for each of the averaged sum rates.

The scheme of "no optimization (F)" means that both the source matrices, $\mathbf{R}_{\mathbf{x}_1}$ and $\mathbf{R}_{\mathbf{x}_2}$, and the relay matrix $\mathbf{F}$ were chosen randomly but meet power constraints at source and relay. The scheme of "no optimization (A)" means that the relay matrix $\mathbf{F}$ meets the optimal structure (2.11) but otherwise is randomly chosen. For "no optimization (F)" and "no optimization (A)", the same source matrices were used. The scheme of "source only (F)" means that the source matrices were optimized but the relay matrix $\mathbf{F}$ was chosen as in "no optimization(F)". The scheme of "source only (A)" means that the source matrices were optimized and $\mathbf{A}$ was chosen as in "no optimization(A)". The scheme of "relay only" means that the source matrices were chosen as in "no optimization" but the relay matrix $\mathbf{F}$ was completely optimized. The scheme of "joint source-relay optimization" means that both the source matrices and the relay matrix were optimized alternately until convergence.

For the joint optimization, about 5-10 alternations were needed until convergence

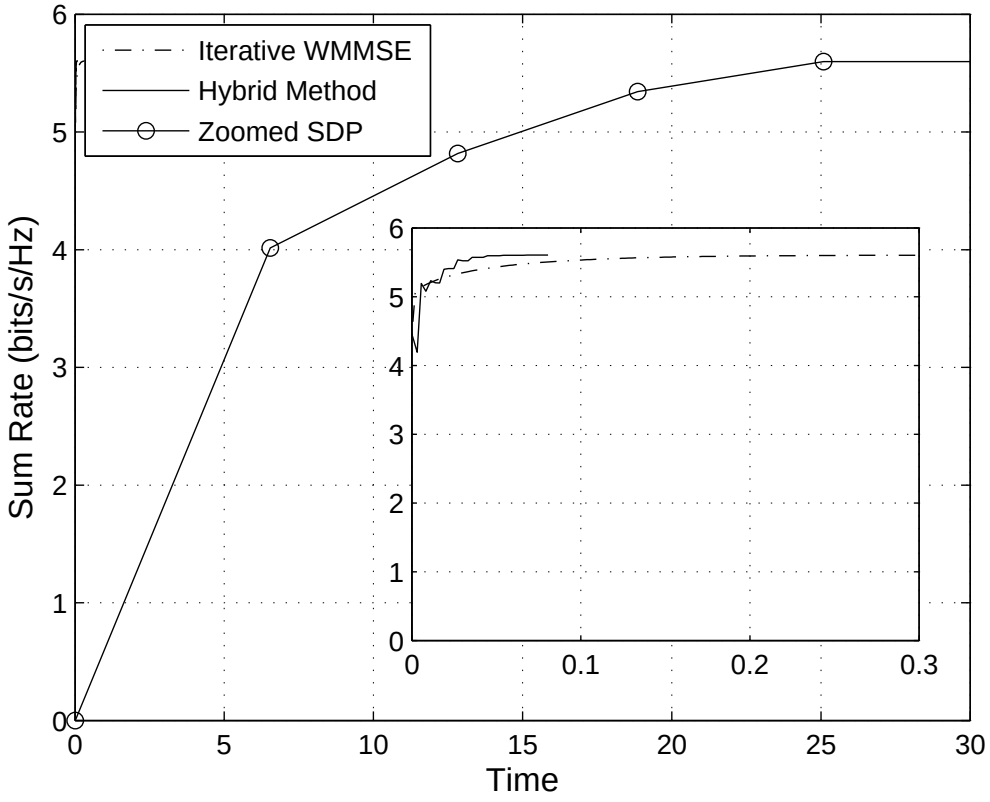

Figure 2.4: The time-capacity curves of the zoomed SDP method based on [77], the hybrid gradient method and the iterative WMMSE method. $N = 1$ and $M = 5$. The time is in seconds. $\text{SNR}_1 = \text{SNR}_2 = \text{SNR}_R = 10dB$. The zoomed SDP is approximately 1000 times slower than the hybrid gradient.

with the stopping criterion 10^{-3}. Both WMMSE and hybrid gradient methods were used (in separate runs) for finding the relay matrix at each alternation. The same result was obtained. This is because WMMSE and hybrid gradient yielded the same locally converged result.

In all cases, the power constraints at the sources and the relay were met with equality. The order of these curves is as expected. However, the relay only optimization yields the largest gain of the sum rate. This is because the number of antennas at the relay

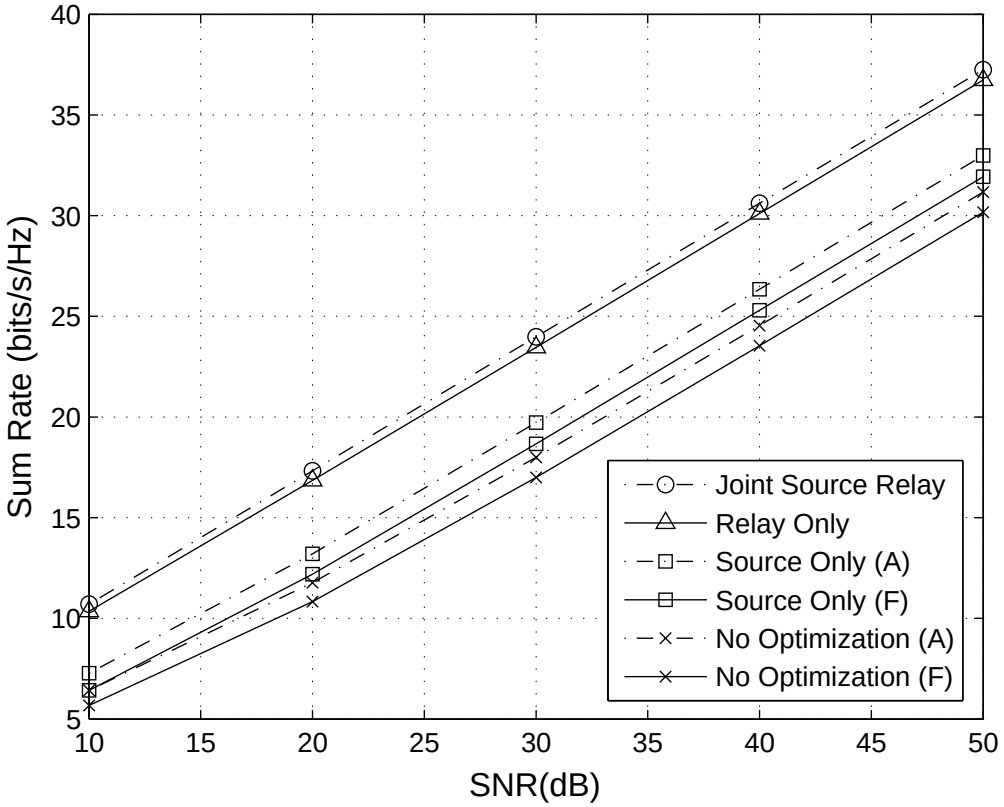

Figure 2.5: Average sum rate under different schemes versus SNR = SNR_1 = SNR_2 = SNR_R. Averaged over 100 randomly generated channels. $N = 2$ and $M = 6$.

(M) is significantly larger the number of antennas at the users (N). Here, $M = 6$ and $N = 2$.

Fig. 2.6 illustrates the effect of the number of antennas on the maximum sum rate under the joint source-relay optimization. We define $\gamma = M/N$ in this figure. We see that both N and M have a significant effect on the sum rate. The effect of M on the sum rate becomes more significant as N becomes larger. This property makes Theorem 2.3.1 important in reducing the complexity.

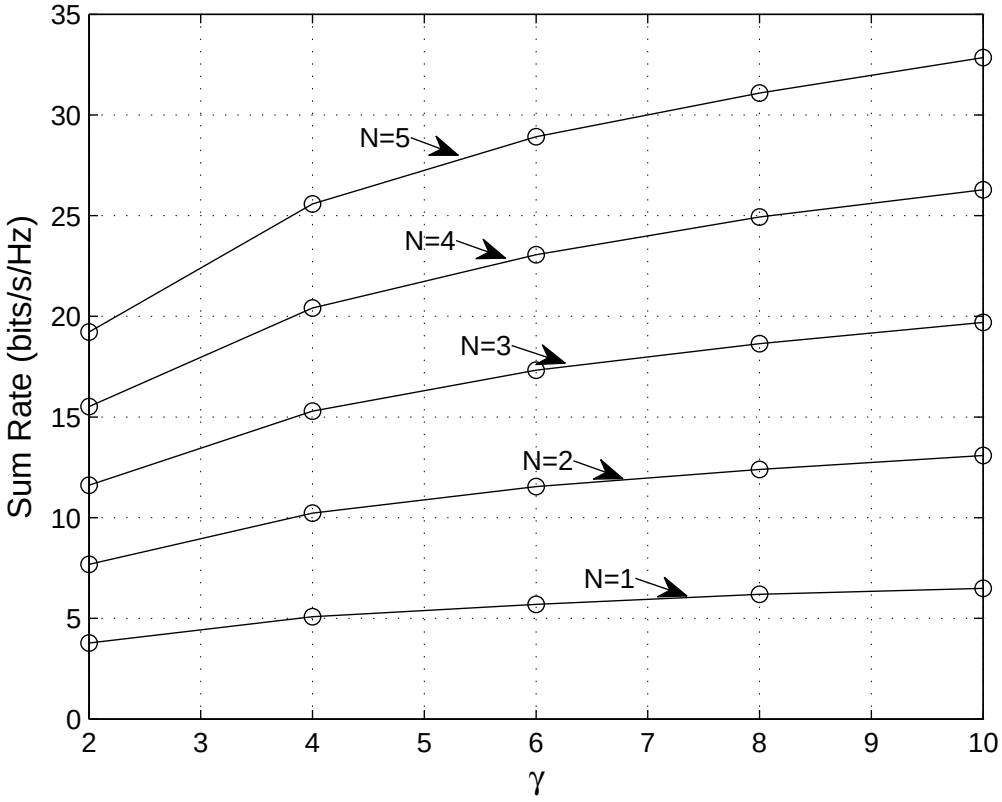

Figure 2.6: Averaged sum rate of the two-way relay system under joint source-and-relay optimization versus $\gamma = M/N$. $\mathrm{SNR}_1 = \mathrm{SNR}_2 = 10dB$ and $\mathrm{SNR}_R = 13dB$.

2.8 Chapter Conclusion

In this chapter, we have shown a study of the optimal design of the source and relay matrices for a non-regenerative two-way MIMO relay system. Although the two-way scheme is spectrally efficient for a half-duplex MIMO relay, the design of the optimal source matrices and the optimal relay matrix is not trivial especially when all nodes have multiple antennas. This study has shown an optimal structure for the relay matrix, which is useful for reducing the complexity of the optimal design when the relay has much more antennas

than the users. For relay matrix optimization, we have developed the hybrid gradient method and the iterative WMMSE method, both of which have consistently yielded the same results. The hybrid gradient method is faster to converge, and the iterative WMMSE method is easier to implement. Both of these methods are much faster than the SDP based optimization method in [77]. The latter is only applicable to single-antenna users. We have established that for any given relay matrix, the optimal design of the source matrices (for uniformly weighted sum rate) follows the generalized water filling algorithm in [76]. We have demonstrated that by alternating the relay optimization and the source optimization, our joint source-and-relay optimization method can yield much improved system capacity especially when the number of antennas at all users is large.

Chapter 3

Thermal Modeling and Temperature Prediction Using Least Square Model Averaging with Model Screening

3.1 Introduction

In this chapter, a simple and perhaps the most basic approach to Black-Box MIMO Thermal Modeling is presented. We use a deterministic MIMO thermal system whose inputs are the powers consumed by the cores in the system and whose outputs are the temperatures at certain predetermined points in the system to model a multicore microprocessor system. The classical Least Square (LS) algorithm with model order selection could be applied to

estimate the parameters of the deterministic MIMO ARMA model that governs the input-output relationship. However, the model uncertainty inherited in model order selection is neglected after the final model is chosen [22]. We develop the model averaging algorithm based on least square with model screening to overcome this uncertainty. The proposed algorithm is shown to provide better prediction performance compared to those with model order selection.

The rest of the chapter is organized as follows. In Section 3.2, we describe the details of the deterministic MIMO ARMA model. In Section 3.3, we present the classical LS algorithm for estimating the model parameters and Akaike Information Criterion (AIC) rule for model order selection. In Section 3.4, a LS based model averaging algorithm with model screening is proposed. Section 3.5 illustrates the performance of LS and the proposed algorithm. All simulation results were based on real data received from Intel.

3.2 Proposed Thermal Model

Denote all powers consumed by the system at time n by the input vector $\mathbf{u}(n) \in \mathbb{R}^{M\times 1}$ and the temperatures at certain points in the system at time n by the output vector $\mathbf{y}_a(n) \in \mathbb{R}^{N\times 1}$. Here, the notation $\mathbb{R}^{l\times k}$ means the set of all $l \times k$ real matrices. The number of inputs, M, may or may not be the same as the number of outputs, N, although one can choose $M = N$ which is the number of cores in the system. Let b be an environment temperature of the system. Then the deviation of $\mathbf{y}_a(n)$ from b is $\mathbf{y}(n) = \mathbf{y}_a(n) - b\mathbf{1}_N$ where $\mathbf{1}_N = [1, \cdots, 1]^T \in \mathbb{R}^{N\times 1}$.

Under a causality condition, we can treat $\mathbf{y}(n)$ as a function of $\mathbf{y}(n-1), \cdots, \mathbf{y}(n-p)$

and $\mathbf{u}(n), \cdots, \mathbf{u}(n-q)$ plus noise, i.e.,

$$\mathbf{y}(n) = \mathbf{f}\left(\mathbf{y}(n-1), \cdots, \mathbf{y}(n-p), \mathbf{u}(n), \cdots, \mathbf{u}(n-q)\right) + \mathbf{e}(n) \tag{3.1}$$

where $\mathbf{e}(n)$ is the noise or model error, p is the regression order with respect to the output, and q is the regression order with respect to the input. Since $\mathbf{y}(n)$ is the deviation of the real temperature from the environment temperature, the function $\mathbf{f}$ should be such that $\mathbf{f}(0, \cdots, 0) = 0$.

In general, the function $\mathbf{f}$ can be nonlinear and time-varying. But within a limited dynamic range and a limited time window, it is reasonable to choose $\mathbf{f}$ to be linear and time-invariant. In this case, we can replace (3.1) by its first order Taylor's series expansion as follows

$$\mathbf{y}(n) = \sum_{i=1}^{p} \mathbf{A}_i^T \mathbf{y}(n-i) + \sum_{i=0}^{q} \mathbf{B}_i^T \mathbf{u}(n-i) + \mathbf{e}(n) \tag{3.2}$$

which is the MIMO ARMA model. The coefficient matrices $\mathbf{A}_i \in \mathbb{R}^{N \times N}$ and $\mathbf{B}_i \in \mathbb{R}^{N \times M}$ may change from one multicore system to another, from one environment to another. The determination of these matrices is known as system identification.

Unlike [13] where the powers are assumed to be unknown but stationary random processes, we treat the powers as deterministic and known quantities. Our assumption is more realistic and allows a more reliable estimation of the ARMA model parameters.

A more compact form of (3.2) is

$$\begin{aligned} \mathbf{y}(n) &= \mathbf{A}^T \mathbf{y}_c(n) + \mathbf{B}^T \mathbf{u}_c(n) + \mathbf{e}(n) \\ &= \mathbf{W}^T \mathbf{z}(n) + \mathbf{e}(n) \end{aligned} \tag{3.3}$$

where $\mathbf{z}(n) = [\mathbf{y}_c^T(n), \mathbf{u}_c^T(n)]^T \in \mathbb{R}^{(pN+(q+1)M)\times 1}$, $\mathbf{y}_c(n) = \left[\mathbf{y}^T(n-1), \cdots, \mathbf{y}^T(n-p)\right]^T$, $\mathbf{u}_c(n) = \left[\mathbf{u}^T(n), \cdots, \mathbf{u}^T(n-q)\right]^T$, $\mathbf{W} = [\mathbf{A}^T, \mathbf{B}^T]^T$, $\mathbf{A} = [\mathbf{A}_1^T, ..., \mathbf{A}_p^T]^T$ and $\mathbf{B} = [\mathbf{B}_0^T, ..., \mathbf{B}_q^T]^T$.

Assuming $\mathbf{e}(n) = 0$ in (3.2), the system's transfer function is given by

$$\mathbf{H}(z) = -\left(\sum_{i=0}^{p} \mathbf{A}_i^T z^{-i}\right)^{-1} \left(\sum_{i=0}^{q} \mathbf{B}_i^T z^{-i}\right) \tag{3.4}$$

where $\mathbf{A}_0 = -\mathbf{I}$.

If the system has a weak non-linearity, we can let the function $\mathbf{f}$ to be approximated by its second-order Taylor series expansion. In this case, we could write the system model as

$$\begin{aligned} \mathbf{y}(n) &= \mathbf{W}^T \mathbf{z}(n) + \mathbf{V}^T (\mathbf{z}(n) \otimes \mathbf{z}(n)) + \mathbf{e}(n) \\ &= \mathbf{W}_c^T \mathbf{z}_c(n) + \mathbf{e}(n) \end{aligned} \tag{3.5}$$

where $\otimes$ denotes the Kronecker product, both $\mathbf{W}^T$ and $\mathbf{V}^T$ are constant matrices, $\mathbf{W}_c^T = (\mathbf{W}^T, \mathbf{V}^T)$, and $\mathbf{z}_c(n)^T = (\mathbf{z}(n)^T, \mathbf{z}(n)^T \otimes \mathbf{z}(n)^T)$. The last expression of (3.3) is identical to that of (3.5) except that the dimension of the latter is much larger. Any algorithm developed for identifying $\mathbf{W}$ in (3.3) can be similarly used for identifying $\mathbf{W}_c$ in (3.5).

However, given the same number of observations of the inputs and outputs of the system, the estimation of $\mathbf{W}$ is generally much more reliable than that of $\mathbf{W}_c$.

However, our simulation based real data shows that the thermal behavior of a multicore microprocessor is described better by the linear MIMO ARMA model (3.3) than by its nonlinear extension (3.5). Therefore, we will not further consider (3.5).

3.3 Least Square (LS) Algorithm with Model Order Selection

Given the thermal model (3.3), an algorithm is needed to estimate the matrix $\mathbf{W}$ by using all available $\mathbf{y}(n)$ and $\mathbf{z}(n)$. Notice that $\mathbf{z}(n)$ is determined by $\mathbf{u}(n), \cdots, \mathbf{u}(n-q)$ and $\mathbf{y}(n-1), \cdots, \mathbf{y}(n-p)$.

The dimension of $\mathbf{W}$ depends not only on M and N (which are given by the problem) but also on the regression orders p and q of the MIMO ARMA model. If p and q are too small, the model is under-parameterized. If p and q are too large, the model is over-parameterized. An under-parameterized model will lead to a large model-mismatch error $\mathbf{e}(n)$. An over-parameterized model often leads to a small model-mismatch error $\mathbf{e}(n)$ but its large number of parameters are difficult to estimate. It is known in statistics that given a fixed amount of observations, the more parameters there are, the less reliable the estimation becomes. In practice, the role of thumb is to choose a just large enough number of parameters to make the model-mismatch small enough. Advanced theories on model order selection are available in [28] and the references therein although they are not immediately

applicable to the thermal modeling applications. In this section, we first assume that p and q have been chosen and then introduce Akaike Information Criterion (AIC), a widely used model order selection rule [57] that will also be used later for model averaging.

Let $\mathbf{y}(n)$ and $\mathbf{z}(n)$ be available for $n = n_1, \cdots, n_2$. We can determine $\mathbf{W}$ by minimizing the following cost

$$\begin{aligned} J^{LS}(\mathbf{W}) &= \sum_{i=n_1}^{n_2} \|\mathbf{y}(i) - \mathbf{W}^T\mathbf{z}(i)\|^2 \\ &= \sum_{i=n_1}^{n_2} \mathrm{Tr}\left([\mathbf{y}(i) - \mathbf{W}^T\mathbf{z}(i)][\mathbf{y}(i) - \mathbf{W}^T\mathbf{z}(i)]^T\right) \end{aligned}$$

which is known as the LS problem. Tr denotes the trace. The LS problem minimizes the model mismatch error $\mathbf{e}(n)$ in (3.2). The solution to this problem can be found by setting the gradient of $J^{LS}(\mathbf{W})$ with respect to $\mathbf{W}$ to zero. To find the gradient, we first take the differential of $J^{LS}(\mathbf{W})$, i.e.,

$$\partial J^{LS}(\mathbf{W}) = -2 \sum_{i=n_1}^{n_2} \mathrm{Tr}\left([\mathbf{y}(i) - \mathbf{W}^T\mathbf{z}(i)]\mathbf{z}(i)^T\partial\mathbf{W}\right) \tag{3.6}$$

where we have applied the identity $\mathrm{Tr}(AB) = \mathrm{Tr}(BA) = \mathrm{Tr}(B^TA^T)$. From this expression, it follows that

$$\frac{\partial J^{LS}(\mathbf{W})}{\partial \mathbf{W}} = -2 \sum_{i=n_1}^{n_2} \mathbf{z}(i)[\mathbf{y}(i) - \mathbf{W}^T\mathbf{z}(i)]^T \tag{3.7}$$

where we have applied that if $\partial J = \mathrm{Tr}(A\partial X)$ then $\frac{\partial J}{\partial X} = A^T$. Setting the gradient to zero

leads to

$$\mathbf{ZW} = \mathbf{Y} \tag{3.8}$$

where $\mathbf{Z} = \sum_{i=n_1}^{n_2} \mathbf{z}(i)\mathbf{z}^T(i)$ and $\mathbf{Y} = \sum_{i=n_1}^{n_2} \mathbf{z}(i)\mathbf{y}^T(i)$. If $\mathbf{Z}$ is invertible which is typically the case when $n_2 - n_1 + 1$ is no less than the dimension $pN + (q+1)M$ of $\mathbf{z}(n)$, then the LS solution is

$$\hat{\mathbf{W}} = \mathbf{Z}^{-1}\mathbf{Y} \tag{3.9}$$

The above expression is also known as the batch LS algorithm where $\mathbf{Z}$, $\mathbf{Y}$ and $\mathbf{Z}^{-1}\mathbf{Y}$ are determined by using all available data at once. This algorithm has the complexity in the order of $(pN + (q+1)M)^3$ multiplications.

To derive AIC rule, we assume that the elements of $\mathbf{e}(n)$ are i.i.d zero mean Gaussian with variance σ^2 for simplicity. Following [57] we have

$$\begin{aligned} \mathrm{AIC}(p,q) &= -2\ln p\big(\mathbf{y}(n_1),\ldots,\mathbf{y}(n_2);\hat{\mathbf{W}}(p,q)\big) \\ &\quad +2N(pN+(q+1)M) \\ &= N(n_2-n_1+1)\left(\ln 2\pi + 1 + \ln\hat{\sigma}^2(p,q)\right) \\ &\quad +2N(pN+(q+1)M) \end{aligned} \tag{3.10}$$

where $p\big(\mathbf{y}(n_1),\ldots,\mathbf{y}(n_2);\hat{\mathbf{W}}(p,q)\big)$ is the probability density function of $\mathbf{y}(n_1),\ldots,\mathbf{y}(n_2)$ given LS estimate $\hat{\mathbf{W}}(p,q)$ and

$$\hat{\sigma}^2(p,q) = \frac{\sum_{i=n_1}^{n_2} \|\mathbf{y}(i) - \hat{\mathbf{W}}(p,q)^T\mathbf{z}(i)\|^2}{N(n_2-n_1+1)} \tag{3.11}$$

Then, the model orders are chosen as the ones minimizing the above AIC among a set of candidate model orders. So far, we are able to obtain the LS estimate of model parameters according to AIC rule from a set of model order candidates. Note that other model order selection rules can also be employed such as Bayesian Information Criterion (BIC) and Generalized cross-validatory KL approach (GIC) [57].

3.4 Least Square Model Averaging with Model Screening

Note that we assume that the model orders are chosen as hopefully the "best" ones by model order selection techniques in the last section. However, it has been recognized that the uncertainty inherited in model order selection is ignored once the "best" model is finalized and the corresponding inference may lead to overly optimistic results [22]. By model order selection, the final model is chosen in terms of some model evaluation criteria and further inference is conducted according to the corresponding conditional statistical characteristics of its parameter estimates. In case that the selected model orders deviate from the true ones, relying on this only model could result in misleading corresponding inference. On the other hand, model averaging, as will be described below, mitigates the dependence on only one selected model through combining estimated model parameters across different model candidates. In fact, by averaging the inherited model uncertainty resulted from model order selection is incorporated rather than neglected.

Unlike in the last section we consider a sequence of different models indexed by $k = 1, \ldots, K$, where the kth model employs p_k and q_k as its output and input model orders.

Following (3.1) those models can be expressed as

$$\mathbf{y}(n) = \mathbf{f}_k\left(\mathbf{y}(n-1), \cdots, \mathbf{y}(n-p_k), \mathbf{u}(n), \cdots, \mathbf{u}(n-q_k)\right) + \mathbf{e}_k(n) \tag{3.12}$$

Similar to (3.3) the compact approximation of the first order Taylor's expansion is given by

$$\mathbf{y}(n) = \mathbf{W}_k^T \mathbf{z}_k(n) + \mathbf{e}_k(n) \tag{3.13}$$

where $\mathbf{z}_k(n) = [\mathbf{y}_{kc}^T(n), \mathbf{u}_{kc}^T(n)]^T \in \mathbb{R}^{(p_k N + (q_k+1)M)\times 1}$, $\mathbf{y}_{kc}(n) = \left[\mathbf{y}^T(n-1), \cdots, \mathbf{y}^T(n-p_k)\right]^T$, $\mathbf{u}_{kc}(n) = \left[\mathbf{u}^T(n), \cdots, \mathbf{u}^T(n-q_k)\right]^T$, $\mathbf{W}_k = [\mathbf{A}_k^T, \mathbf{B}_k^T]^T$, $\mathbf{A}_k = [\mathbf{A}_{k1}^T, ..., \mathbf{A}_{kp_k}^T]^T$ and $\mathbf{B}_k = [\mathbf{B}_{k0}^T, ..., \mathbf{B}_{kq_k}^T]^T$.

In order to combine different models, we define $\mathbf{w} = [w_1, \ldots, w_K]$ as the weight vector where $0 \le w_k \le 1$ and $\sum_{k=1}^K w_k = 1$. The kth element of $\mathbf{w}$ is assigned to the kth model and the new estimated model parameter is given by

$$\hat{\mathbf{W}}_{MA} = \sum_{k=1}^{K} w_k \begin{bmatrix} \hat{\mathbf{A}}_k^T & \mathbf{0} & \hat{\mathbf{B}}_k^T & \mathbf{0} \end{bmatrix}^T \tag{3.14}$$

where $\hat{\mathbf{W}}_{MA}$ is a $p_{\max}N + (q_{\max}+1)M$ by N parameter matrix, $p_{\max}$ and $q_{\max}$ are the maximum among p_k and q_k, $\hat{\mathbf{A}}_k$ and $\hat{\mathbf{B}}_k$ result from the LS estimate of $\hat{\mathbf{W}}_k$ by (3.9) based on the kth model and the zero elements in $\hat{\mathbf{W}}_{MA}$ are assigned to those data that don't appear in models with smaller p_k, q_k but do exist in larger orders to keep this consistent formulation. Since the LS algorithm provides the solution of $\hat{\mathbf{W}}_k$, the only concern left is how to determine the weight vector $\mathbf{w}$.

For this problem, we first compute the AIC_k for each model as in (3.10) and then

choose a number of $K' < K$ models with the smallest AIC values to screen out those less matching candidates. Let Γ denote the set of selected model candidates. Then, smoothed Akaike Information Criterion (S-AIC), a simplified form of Akaike model averaging [22], is adopted where the weight associated with the kth model in Γ is obtained as

$$w_k = \frac{\exp\left(-\frac{1}{2}\mathrm{AIC}_k\right)}{\sum_{k\in\Gamma}\exp\left(-\frac{1}{2}\mathrm{AIC}_k\right)}, \quad k \in \Gamma \tag{3.15}$$

The proposed least square model averaging (LSMA) with model screening algorithm is illustrated in Algorithm 3.1.

Algorithm 3.1 (LS Model Averaging with Model Screening (LSMA))**.**

1: *Given p_k, q_k for each model k, $k = 1, \ldots, K$, prepare data $\mathbf{Z}_k$ and $\mathbf{Y}_k$.*

2: *Estimate $\hat{\mathbf{W}}_k$ using LS algorithm of* (3.9) *for each candidate model and compute the estimate ${\hat{\sigma}_i}^2$ of σ_i^2 by* (3.11)*.*

3: *Calculate AIC by* (3.10) *for each model k. Select $K' < K$ models with the smallest AIC values and let Γ denote the set of these models.*

4: *Compute the K'-dimensional weight vector $\mathbf{w}$ by* (3.15)*. Note that $\sum_{k\in\Gamma} w_k = 1$.*

5: *Combine the K' models as in* (3.14) *to obtain the LS model averaging estimate of model parameters.*

3.5 Simulation Results

To verify the proposed thermal model and estimation algorithm, we use the real data for a quad-core system with one cache core obtained from Intel as shown in Fig. 3.1

and Fig. 3.2 where $M = 5$ and $N = 5$. The whole set of data is divided into two parts. Training part comprises the first 1000 samples for the estimation of model parameters and validation part comprises the second 1000 samples for measuring the accuracy of prediction performance. For the data we use, the sampling (or iteration) interval is uniform and equals 0.001s.

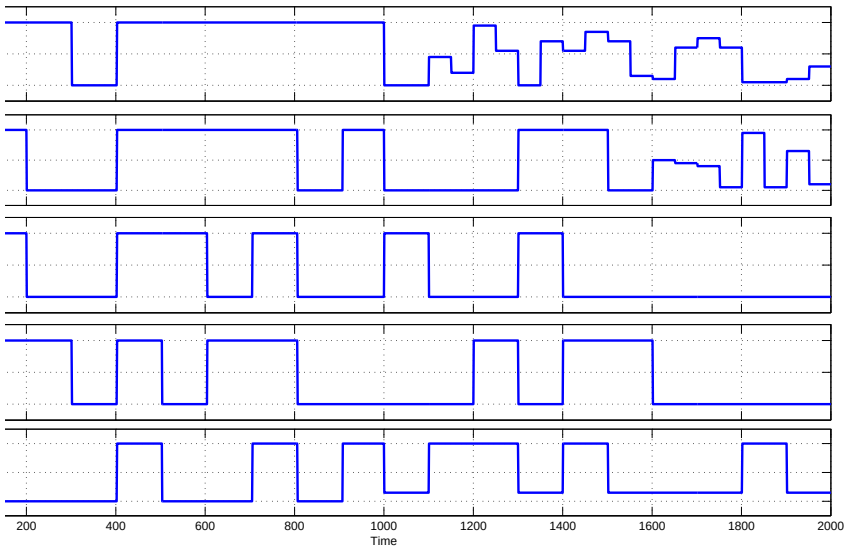

Figure 3.1: Power inputs, $M = 5$.

Our simulation results show that although the choice of the ambient temperature b affects the results of the estimated MIMO ARMA parameters in $\mathbf{W}$, it has little effect on the modeling accuracy in terms of the model mismatch $\mathbf{e}(n)$. In other words, when we tried to estimate both b and $\mathbf{W}$ jointly in the LS fitting, we found that the LS cost is almost invariant to a large range of b and $\mathbf{W}$. From a practical point of view, it is a good news that the MIMO ARMA modeling is not sensitive to b. We will choose $b = 30$ which is less

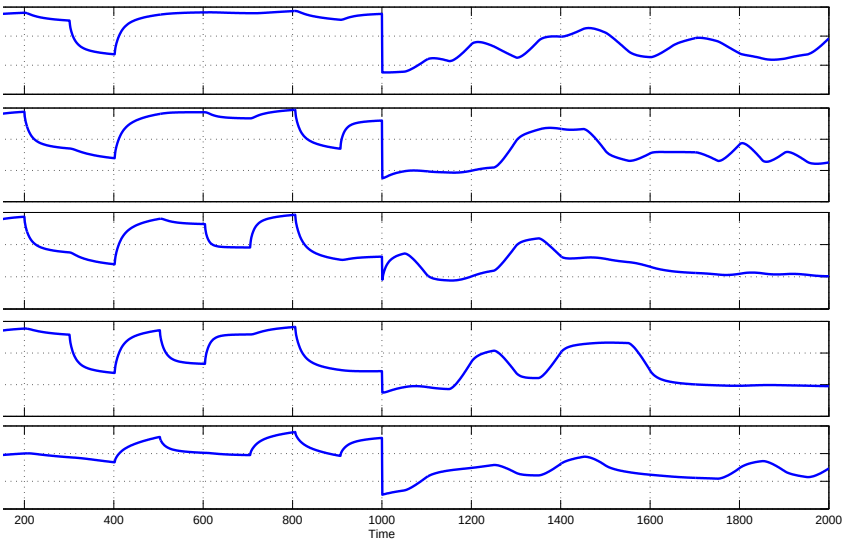

Figure 3.2: Temperature outputs in Celsius, $N = 5$.

than the minimum value in Fig. 3.2.

For simplicity, we consider the set of $K = 10$ models with model orders $p_k = q_k = k$, $k = 1, \ldots, 10$. For each model, the LS algorithm uses the 1000 training data to determine a model matrix $\hat{\mathbf{W}}$ via the batch LS algorithm. The model mismatch error due to $\hat{\mathbf{W}}$ for the entire data is defined as

$$\hat{\mathbf{e}}(i) = \mathbf{y}(i) - \hat{\mathbf{W}}^T \mathbf{z}(i) \tag{3.16}$$

The corresponding RMSE (root mean squared error) of the model mismatch for the training data as the estimation error is

$$e_T = \sqrt{\frac{\sum_{i=k+1}^{n_T} \|\hat{\mathbf{e}}(i)\|^2}{(n_T - k)N}} \tag{3.17}$$

where $n_T = 1000$. Note that from the definition of $\mathbf{z}(i)$ and the available data set, $\mathbf{z}(i)$ is

available starting from $i = \max(p_k, q_k) + 1$. Similarly, the RMSE of the model mismatch due to $\hat{\mathbf{W}}$ for the validation data as the prediction error is defined as

$$e_V = \sqrt{\frac{\sum_{i=n_T+k+1}^{n_V} \|\hat{\mathbf{e}}(i)\|^2}{(n_V - n_T - k)N}} \tag{3.18}$$

where $n_V = 2000$.

To evaluate the performance of proposed algorithms, we choose the model orders corresponding to the 1st (model 7), 2nd (model 6) and 9th (model 2) smallest AIC values. From Fig. 3.3 we can see that model order selection does have a significant impact on thermal modeling and temperature prediction due to different model mismatches under different chosen model orders as shown in Table 3.1. However, model 7 with the smallest AIC has a larger prediction error $e_V = 0.2590$ than model 6 with $e_V = 0.2503$ and the 2nd smallest AIC. In case that model 7 is used for future temperature prediction, it may not perform as well as model 5 for prediction, which justifies the existence of uncertainty from model order selection.

Table 3.1: e_T and e_V for Model 7, Model 6, Model 2 and LSMA.

	Model 7	Model 6	Model 2	LSMA
e_T	0.0003	0.0003	0.0072	0.0003
e_V	0.2590	0.2503	0.3475	0.2570

Although we know that for the current data, model 6 has better prediction performance than model 7 with the smallest AIC value, in general which model performs the best has to be decided after we finish data validation. Thus, before any validation data

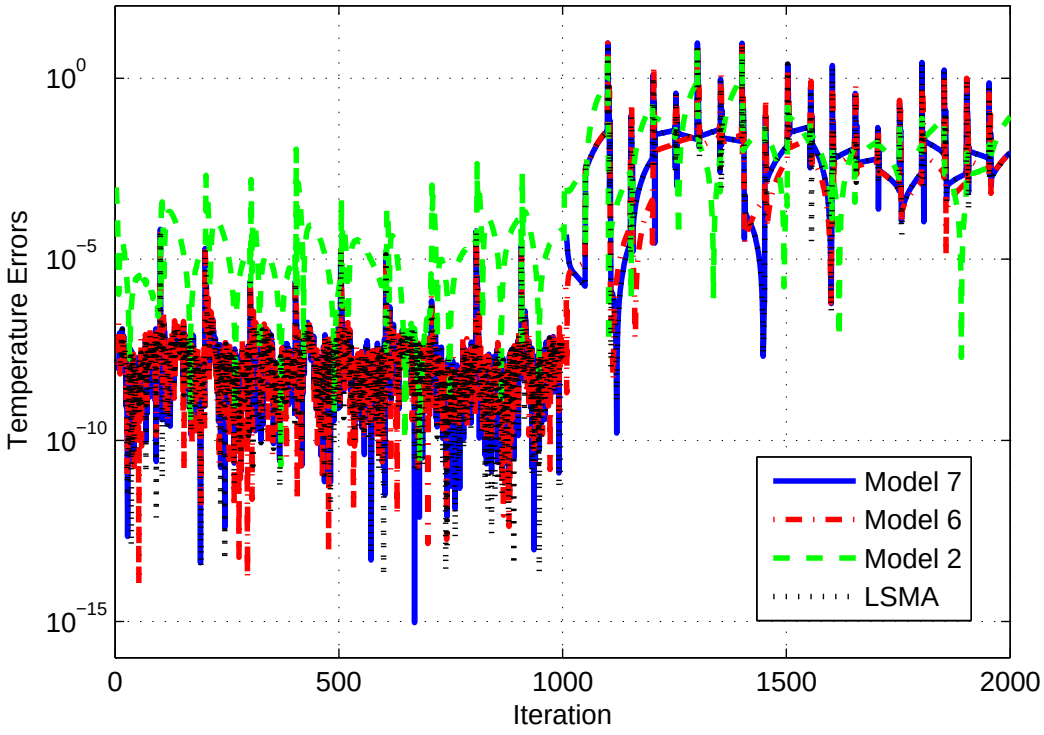

Figure 3.3: Model-mismatch errors for output 3 of model 7, model 6, and model 2.

are available, LS model averaging with model screening of Algorithm 3.1 alleviates the uncertainty resulted from model order selection. Let assume $K' = 3$, then the K' smallest AIC values have model index 7, 6, and 8, respectively, and their corresponding weights can be computed as $\mathbf{w} = [0.867, 0.071, 0.062]^T$. Since most model mismatches of the new parameters estimated by Algorithm 3.1 are so close to that of individual models that they can not almost be distinguished, we just show its estimation and prediction errors in Table 3.1 without displaying its model mismatch curve in Fig. 3.3. We can see that the prediction error from Algorithm 3.1 is 0.2570 which is smaller than that of model 7 with the smallest

AIC.

3.6 Conclusion

In this chaper we propose a black-box LTI MIMO thermal model for multicore microprocessor system with power consumptions as inputs and temperature observations as outputs. First, LS method combined with model order selection rules such as AIC is used to estimate the model parameters and furthermore predict model temperatures. Then, in order to incorporate the uncertainty inherited in model order selection, we introduce the idea of model averaging and propose LS based algorithm with model screening whose model weights can be computed by S-AIC. Simulation results show that the proposed algorithm has better prediction performance compared to LS method with AIC. Note that model averaging algorithm can also be extended to online estimation of model parameters by Recursive Least Square (RLS) method.

Chapter 4

Fast Subspace Tracking Algorithms with Multiple Inner Iterations and Ritz Acceleration

4.1 Introduction

While most of the existing subspace tracking algorithms focus on reducing the computational complexity, the convergence rates as well as tracking accuracy measured by subspace distance seem to reach the performance limits by only allowing a single inner iteration at each data update. Intuitively, in order to further improve the algorithm performance, multiple inner iterations can be applied to the approximated data matrix considered as fixed at each data update in terms of classical bi-iteration SVD based algorithms [58] [3] [44]. In this hapter following this idea, we find that the performance improvement could

be marginal so that the increase of computational loads is not worthwhile. This is because during the transition period both old and new principal subspaces are contained in the data matrix so that the actual dimension of the underlying principal subspace is doubled. Thus, the approximated data matrix cannot contain most of the information by using the lower-rank consistent approximation whose principal subspace dimension is only half of the actual dimension. This inaccurate approximation leads to the desire of tracking a principal subspace with an enlarged dimension. On the other hand, the introduction of multiple inner iterations at each data update also inevitably increases the algorithm complexity. In order to achieve better performance with a smaller number of inner iterations, Ritz acceleration is employed to further improve the convergence rate and tracking accuracy of the proposed algorithm where the SVD of a much smaller-dimensional matrix is computed with only a small amount of extra computations given $N \gg r$. Since existing algorithms fail to further improve the convergence rate and tracking accuracy, algorithm complexity has to be increased to achieve better performance. Under such a premise we will show that the proposed algorithms with multiple inner iterations and Ritz acceleration accomplish much better performance in both aspects with already reduced algorithm complexity enhancement.

The rest of this chapter is structured as follows. Section 4.2 gives a review of bi-iteration SVD algorithm and its extensions with multiple inner iterations and Ritz acceleration as the fundamental basis for fast algorithms developed later. The reason to choose an enlarged tracking dimension is also explained. Then, new subspace tracking algorithms are proposed with a lower-rank consistent approximation matrix of an enlarged dimension

in Section 4.4. The ultrafast versions of the above algorithms are also provided. Section 4.5 shows the simulation results for the proposed algorithms and conclusions are made in Section 4.6.

4.2 A Review of Bi-Iteration SVD Algorithm and its Extensions by Ritz Acceleration and Multiple Inner Iterations

First, we consider a fixed data matrix $\mathbf{X} \in \mathbb{C}^{L\times N}$. Let its Singular Value Decomposition (SVD) be $\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^H$ where $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_{r_{max}})$, $|\lambda_1| \geq \ldots \geq |\lambda_{r_{max}}|$, and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_{r_{max}}] \in \mathbb{C}^{L\times r_{max}}$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_{r_{max}}] \in \mathbb{C}^{N\times r_{max}}$ have orthonormal columns with the ith column associated with λ_i, $r_{max} \triangleq \min(L, N)$. It can also be assumed that $L \geq N$, then $r_{max} = N$. We are interested in determining $\mathbf{U}_r = [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{C}^{N\times r}$, i.e. the r principal right singular vectors of $\mathbf{X}$ or alternatively the r principal eigenvectors vectors of $\mathbf{X}^H\mathbf{X}$, where r is usually much smaller than r_{max}. Similar definition could be made about $\mathbf{V}_r = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$ and the singular values associated with $\mathbf{U}_r$ and $\mathbf{V}_r$, $\lambda_1, \ldots, \lambda_r$, are principal singular values. On the other hand, $\mathbf{U}_{r\perp} = [\mathbf{u}_{r+1}, \ldots, \mathbf{u}_{r_{max}}]$, $\mathbf{V}_{r\perp} = [\mathbf{v}_{r+1}, \ldots, \mathbf{v}_{r_{max}}]$ and $\lambda_{r+1}, \ldots, \lambda_{r_{max}}$ are denoted as the minor right and left singular vectors and minor singular values. If $|\lambda_r| \geq |\lambda_{r+1}|$, then the subspace spanned by the columns of $\mathbf{U}_r$, $\mathcal{R}(\mathbf{U}_r)$, is said to be the unique principal invariant subspace of $\mathbf{X}^H\mathbf{X}$ associated with its eigenvalues $|\lambda_1|^2, \ldots, |\lambda_r|^2$. In order to obtain $\mathbf{U}_r$, bi-iteration SVD algorithm described in Algorithm 4.1 [58] [3] can be applied as an alternative of orthogonal iteration [18] with respect to $\mathbf{X}^H\mathbf{X}$, where k denotes the index of iterations.

Algorithm 4.1 (Bi-Iteration SVD)**.**

1: ***Initialization:*** $\mathbf{Q}_A(0) \in \mathbb{C}^{N\times r}$, *and* $\mathbf{Q}_A^T(0)\mathbf{Q}_A(0) = I_r$.

2: ***for*** $k = 1, 2, \ldots$ ***do***

3: $\quad \mathbf{B}(k) \triangleq \mathbf{X}\mathbf{Q}_A(k-1)$

4: $\quad \mathbf{B}(k) = \mathbf{Q}_B(k)\mathbf{R}_B(k)$, *QR decomposition*

5: $\quad \mathbf{A}(k) \triangleq \mathbf{X}^H\mathbf{Q}_B(k)$

6: $\quad \mathbf{A}(k) = \mathbf{Q}_A(k)\mathbf{R}_A(k)$, *QR decomposition*

7: ***end for***

Then, $\mathbf{Q}_A(k) \in \mathbb{C}^{N\times r}$ will converge to $\mathbf{U}_r$ and $\mathbf{Q}_B(k) \in \mathbb{C}^{L\times r}$ will converge to $\mathbf{V}_r = [\mathbf{v}_1, \ldots, \mathbf{v}_r] \in \mathbb{C}^{N\times r}$ as the r principal left singular vectors of $\mathbf{X}$ or alternatively the r principal eigenvectors vectors of $\mathbf{X}\mathbf{X}^H$. To evaluate the difference from two same dimensional subspaces $\mathcal{Y}$ and $\mathcal{Z}$, we employ the distance between two subspaces [18] as

$$dist\,(\mathcal{Y}, \mathcal{Z}) = \|\mathbf{Y}_1\mathbf{Y}_1^H - \mathbf{Z}_1\mathbf{Z}_1^H\|_2 = \|\mathbf{Y}_1^H\mathbf{Z}_2\|_2 = \|\mathbf{Z}_1^H\mathbf{Y}_2\|_2 \tag{4.1}$$

where $\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2] \in \mathbb{C}^{n\times n}$ and $\mathbf{Z} = [\mathbf{Z}_1, \mathbf{Z}_2] \in \mathbb{C}^{n\times n}$ are n by n orthonormal matrices and $\mathcal{Y} = \mathcal{R}(\mathbf{Y}_1)$, $\mathcal{Z} = \mathcal{R}(\mathbf{Z}_1)$. The distance between subspaces also relates to the maximum principal angle θ_{max} by

$$dist\,(\mathcal{Y}, \mathcal{Z}) = \sin(\theta_{max}). \tag{4.2}$$

Then, with respect to distance between subspaces we can have the following relationship between the subspaces spanned by the columns of $\mathbf{Q}_A(k)$ and $\mathbf{U}_r$ under some mild conditions

from [18, Theorem 8.2.2]

$$dist\left(span\left(\mathbf{q}_{A1}^{(k)},\ldots,\mathbf{q}_{Ai}^{(k)}\right),span\left(\mathbf{u}_1,\ldots,\mathbf{u}_i\right)\right)=O\left(\left|\frac{\lambda_{i+1}}{\lambda_i}\right|^{2k}\right),i=1,\ldots,r \tag{4.3}$$

where $\mathbf{Q}_A(k)=\left[\mathbf{q}_{A1}^{(k)},\ldots,\mathbf{q}_{Ar}^{(k)}\right]$. Similar statement can be made for $\mathbf{Q}_B(k)$, too. Furthermore, $\mathbf{R}_A(k)$, $\mathbf{R}_B(k)$ both converge to $\mathbf{D}_r=diag(\lambda_1,\ldots,\lambda_r)$ including the r principal singular values of $\mathbf{X}$ or the square roots of the r principal eigenvalues of $\mathbf{X}^H\mathbf{X}$ or $\mathbf{X}\mathbf{X}^H$.

For adaptive signal processing applications, the hybrid data window to update the time-varying data matrix $\mathbf{X}(t)$ as in [44] is used

$$\begin{bmatrix}\mathbf{X}(t)\\ \alpha^{L/2}\beta^{1/2}\mathbf{x}^H(t-L)\end{bmatrix}=\begin{bmatrix}\beta^{1/2}\mathbf{x}^H(t)\\ \alpha^{1/2}\mathbf{X}(t-1)\end{bmatrix} \tag{4.4}$$

where $\mathbf{X}(t)\in\mathbb{C}^{L\times N}$, $\mathbf{x}(t)\in\mathbb{C}^{N\times 1}$ is the new data vector available at time instant t, and $0<\alpha,\beta\leq 1$. This definition avoids the infinite increase of dimension of $\mathbf{X}(t)$ in [58]. Note that if $\alpha=\beta=1$, (4.4) reduces to the sliding window in [3]. We can also choose $\beta=1-\alpha$ for a sliding exponential window. Let $\mathbf{C}(t)=E\left[\mathbf{x}(t)\mathbf{x}^H(t)\right]$ be the covariance matrix of $\mathbf{x}(t)$, also varying with time. Then, given $\mathbf{X}(t-1)$ and $\mathbf{x}(t)$ its estimate $\hat{\mathbf{C}}(t)$ could be expressed by

$$\hat{\mathbf{C}}(t)=\mathbf{X}^H(t)\mathbf{X}(t)=\alpha\hat{\mathbf{C}}(t-1)+\beta\mathbf{x}(t)\mathbf{x}^H(t)-\alpha^L\beta\mathbf{x}(t-L)\mathbf{x}^H(t-L). \tag{4.5}$$

To track the r principal right singular vectors of $\mathbf{X}(t)$, $\mathbf{X}$ could be replaced by $\mathbf{X}(t)$ and iteration index k by time instant t in Algorithm 4.1. Note that one of the advantages of

bi-iteration SVD algorithm over orthogonal iteration is that data matrix is processed directly without being squared. Thus, numerical problems resulted from ill-conditioned data matrix could be mitigated to some extent. More importantly, in some high-performance subspace tracking algorithms on which the proposed algorithms are based, $\mathbf{X}(t)$ will be further replaced by a lower-rank consistent approximation matrix to reduce algorithm complexity as described in Section 4.4. When applying the same approximation concept to varying data matrix $\mathbf{X}^H(t)\mathbf{X}(t)$ used for orthogonal iteration, the *positive definite* property of $\mathbf{X}^H(t)\mathbf{X}(t)$ will be lost and the corresponding poor approximation matrix may lead to unstable algorithms which are undesired.

4.2.1 Choice of the Actual Dimension of Tracked Subspace

Then, we take a more careful look at how to choose the actual dimension of subspace tracked for $\mathbf{X}(t)$ and its impact on the performance of tracking algorithms. Assume that we desire to track the r principal right singular vectors of $\mathbf{X}(t)$. While r is the actual dimension of subspace tracked in some algorithms [3] [44] [2], the author of [58] pointed out that the necessary subspace dimension should be increased by a factor of 2 around the segment boundaries where the underlying principal subspace from which $\mathbf{x}(t)$ is structured is changing. To further explain this statement, we assume that $\ldots, \mathbf{x}(t_c-2), \mathbf{x}(t_c-1)$ are generated along with the same principal subspace $\mathcal{U}_1$ and at time instant t_c the underlying principal subspace changes. Correspondingly, $\mathbf{x}(t_c), \mathbf{x}(t_c+1), \ldots$ are obtained in terms of another principal subspace $\mathcal{U}_2$. Then, according to (4.4) during the transition period the data matrices $\mathbf{X}(t_c), \ldots, \mathbf{X}(t_c+L-2)$ are contained in the union of the two subspaces

$\mathcal{U}_1 \cup \mathcal{U}_2$ of the dimension as large as $2r$ when there is no intersection between $\mathcal{U}_1$ and $\mathcal{U}_2$. If β is small, the impact of $\mathcal{U}_2$ in $\mathbf{X}(t)$ starting at $t = t_c$ is weak and becomes stronger as t increases while $\mathcal{U}_1$ loses its dominance.

As will be shown later as in [58] [3] [44] [2], a lower-rank consistent approximation matrix of r columns $\hat{\mathbf{X}}(t-1)$ will replace $\mathbf{X}(t-1)$ at each time instant for low complexity algorithms in (4.4). Thus, around the segment boundaries the corresponding estimate $\hat{\mathbf{X}}(t)$ of rank r containing only r estimated principal left singular vectors may not be accurate enough providing that $\mathbf{X}(t)$ actually has a number of $2r$ principal left singular vectors whose associated singular values are obviously larger that the others, especially around the middle of transition period when $\mathcal{U}_1$ and $\mathcal{U}_2$ have almost equivalent impacts on $\mathbf{X}(t)$. Therefore, in terms of (4.3) although the convergence rate of the desired r-dimensional principal subspace for r-dimensional and $2r$-dimensional tracking both equal to $(|\lambda_{r+1}(t)|/|\lambda_r(t)|)^{2k}$, $2r$-dimensional tracking algorithms produce better performance as the increased computations by enlarging r to $2r$ are consumed on a more accurate data matrix approximation of rank $2r$. Note that although $2r$-dimensional subspace is tracked, only r of them are of interest. Meanwhile, since more than r principal singular vectors exist in $\mathbf{X}(t)$ and its estimate $\hat{\mathbf{X}}(t)$, λ_{r+1} does not serve as a minor singular value any more. As a consequence, for $t = t_c, \ldots, t_c + L - 2$, $|\lambda_{r+1}(t)|/|\lambda_r(t)|$ is not as small as that for $t < t_c$, especially during the middle of transition period when $\lambda_{r+1}(t)$ could be rather close to $\lambda_r(t)$, so that the convergence becomes quite slow. Therefore, although providing a more accurate data matrix estimate, the additional complexity imposed by tracking a larger dimensional subspace fails to improve the convergence performance of tracking algorithms. In the next part, Ritz

acceleration is introduced and only a few extra computations will be involved to make the best use of the already increased computations by doubling the tracking dimension.

4.2.2 Ritz Acceleration

Algorithm 4.2 (Bi-Iteration SVD with Ritz Acceleration)**.**

1: ***Initialization:*** $\mathbf{Q}_A(0) \in \mathbb{C}^{N\times r'}$, *and* $\mathbf{Q}_A^H(0)\mathbf{Q}_A(0) = I_{r'}$.

2: ***for*** $k = 1, 2, \ldots$ ***do***

3: $\mathbf{B}(k) \triangleq \mathbf{X}\mathbf{Q}_A(k-1) = \mathbf{X}\bar{\mathbf{Q}}_A(k)\mathbf{U}$

4: $\mathbf{B}(k) = \mathbf{Q}_B(k)\mathbf{R}_B(k)$, *QR decomposition*

5: $\mathbf{A}(k) \triangleq \mathbf{X}^H\mathbf{Q}_B(k)$

6: $\mathbf{A}(k) = \bar{\mathbf{Q}}_A(k)\mathbf{R}_A(k)$, *QR decomposition*

7: $\mathbf{T}(k) \triangleq \bar{\mathbf{Q}}_A^H(k)\left(\mathbf{X}^H\mathbf{X}\right)\bar{\mathbf{Q}}_A(k)$ *or* $\mathbf{X}\bar{\mathbf{Q}}_A(k)$

8: $\mathbf{T}(k) = \mathbf{U}(k)\mathbf{\Sigma}(k)\mathbf{U}^H(k)$ *or* $\mathbf{V}(k)\mathbf{\Lambda}(k)\mathbf{U}^H(k)$, *ED or SVD*

9: $\mathbf{Q}_A(k) = \bar{\mathbf{Q}}_A(k)\mathbf{U}(k)$, *Ritz vectors*

10: *Let* $\mathbf{Q}(k)$ *include the first* r *columns of* $\mathbf{Q}_A(k)$

11: ***end for***

Following the last part assume that the dimension of estimated subspace is enlarged to $r' > r$. To improve the convergence performance, we introduce Ritz acceleration [18] [55] with a faster convergence rate for fixed data matrix $\mathbf{X}$ in Algorithm 4.2 where $\mathbf{Q}(k) \in \mathbb{C}^{N\times r}$ is considered as the estimate for $\mathbf{U}_{r'}$. The extra steps of matrix multiplication, Step 7, and EigenDecomposition (ED) or SVD, Step 8, in Algorithm 4.2 are adopted to further rotate $\bar{\mathbf{Q}}_A(k) \in \mathbb{C}^{N\times r'}$ toward the principal eigenvectors of $\mathbf{X}^H\mathbf{X}$ by a unitary matrix

$\mathbf{U}(k) \in \mathbb{C}^{r' \times r'}$. Ideally, $\mathbf{T}(k)$ contains the r' principal eigenvalues or singular values with a lower rank of r'. Since $r' \ll N$, the ill-condition problem is not much concerned for $\mathbf{T}(k)$, so both ED and SVD could be used for computing $\mathbf{U}(k)$. Also, the computations of the extra steps are mainly contributed from the matrix multiplication of (7) rather than the ED or SVD of (8), when $N \gg r'$ as desired for most applications. As a consequence, the distance between the subspace spanned by the columns of $\mathbf{Q}_A(k)$ and $\mathbf{U}_r$ could be given by [55]

$$dist\left(span\left(\mathbf{q}_{A1}^{(k)}, \ldots, \mathbf{q}_{Ai}^{(k)}\right), span\left(\mathbf{u}_1, \ldots, \mathbf{u}_i\right)\right) = O\left(\left|\frac{\lambda_{r'+1}}{\lambda_i}\right|^{2k}\right), i = 1, \ldots, r' \qquad (4.6)$$

where $\mathbf{Q}_A(k) = \left[\mathbf{q}_{A1}^{(k)}, \ldots, \mathbf{q}_{Ar'}^{(k)}\right]$. If the r-dimensional principal subspace is desired, the accelerated convergence rate of $(|\lambda_{r'+1}|/|\lambda_r|)^{2k}$ is smaller than $(|\lambda_{r+1}|/|\lambda_r|)^{2k}$ for Algorithm 4.1. Again, when applying to adaptive signal processing, $\mathbf{X}$ will be replaced by $\mathbf{X}(t)$ and iteration index k by time instant t. If we let $r' \geq 2r$, then during the transition period $\lambda_{r'+1}(t)$ as a minor singular value is much smaller than $\lambda_{r+1}(t)$ that could actually be a principal singular value due to the increased number of dimension of principal subspace involved in the data matrix $\mathbf{X}(t)$. In fact, when $\mathbf{X}(t)^H\mathbf{X}(t)$ or its modification is regarded to approximate the covariance matrix $\mathbf{C}(t)$, the spans of $\mathbf{Q}_A(t)$ or $\bar{\mathbf{Q}}_A(t)$ and corresponding $\mathbf{Q}(t)$ are desired as the principal subspace of $\mathbf{C}(t)$ while $\mathbf{Q}_B(t)$ is just an intermediate matrix not of interest. For fixed data matrix, the diagonal elements in $\mathbf{\Sigma}(k)$ and the columns in $\mathbf{Q}_A(k)$ are called the Ritz values and vectors of $\mathbf{X}^H\mathbf{X}$ at the kth iteration. Thus, the convergence rate of tracking algorithms could be improved by integrating Ritz acceleration into bi-iteration SVD in adaptive signal processing applications.

4.2.3 Multiple Inner Iterations

In the current proposed subspace tracking algorithm with $\mathbf{X}$ replaced by $\mathbf{X}(t)$ and k by t for Algorithm 4.2, only one iteration is operated for each data update at time instant t. Intuitively, in order to further improve the tracking accuracy at each t, multiple inner iterations along with Ritz acceleration can be taken for the same $\mathbf{X}(t)$ as described in Algorithm 4.3. Note that in this proposed algorithm k is no longer the iteration index in Algorithm 4.1 and 4.2 but the index of inner iteration for each date update at time instant t. K is the total number of inner iterations for each t. If we let $K = 1$, the above algorithm is equivalent to Algorithm 4.2 with $\mathbf{X}$ replaced by $\mathbf{X}(t)$ and k by t.

Algorithm 4.3 (Bi-Iteration SVD with Ritz Acceleration and Multiple Inner Iterations for Subspace Tracking)**.**

1: ***Initialization:*** $\bar{\mathbf{Q}}_A^{(K)}(0) \in \mathbb{C}^{N\times r'}$, *and* $\bar{\mathbf{Q}}_A^{(K)H}(0)\bar{\mathbf{Q}}_A^{(K)}(0) = I_{r'}$, $\mathbf{R}^{(K)}(0) = I_{r'}$, $\mathbf{U}^{(K)}(0) = I_{r'}$, $\mathbf{T}^{(K)}(0) = \mathbf{0}_{r'}$.

2: ***for*** $t = 1, 2, \ldots$ ***do***

3: *Let* $\bar{\mathbf{Q}}_A^{(0)}(t) = \bar{\mathbf{Q}}_A^{(K)}(t-1)$, $\mathbf{U}^{(0)}(t) = \mathbf{U}^{(K)}(t-1)$.

4: ***for*** $k = 1, 2, \ldots, K$ ***do***

5: $\mathbf{B}^{(k)}(t) \triangleq \mathbf{X}(t)\mathbf{Q}_A^{(k-1)}(t) = \mathbf{X}(t)\bar{\mathbf{Q}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t)$

6: $\mathbf{B}^{(k)}(t) = \mathbf{Q}_B^{(k)}(t)\mathbf{R}_B^{(k)}(t)$, *the first QR decomposition*

7: $\mathbf{A}^{(k)}(t) \triangleq \mathbf{X}^H(t)\mathbf{Q}_B^{(k)}(t)$

8: $\mathbf{A}^{(k)}(t) = \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{R}_A^{(k)}(t)$, *the second QR decomposition*

9: $\mathbf{T}^{(k)}(t) \triangleq \bar{\mathbf{Q}}_A^{(k)H}(t)\left(\mathbf{X}^H(t)\mathbf{X}(t)\right)\bar{\mathbf{Q}}_A^{(k)}(t)$ *or* $\mathbf{X}(t)\bar{\mathbf{Q}}_A^{(k)}(t)$

10: $\mathbf{T}^{(k)}(t) = \mathbf{U}^{(k)}(t)\mathbf{\Sigma}(k)^{(k)}(t)\mathbf{U}^{(k)H}(t)$ *or* $\mathbf{V}^{(k)}(t)\mathbf{\Lambda}(k)^{(k)}(t)\mathbf{U}^{(k)H}(t)$, *ED or SVD*

11: $\mathbf{Q}_A^{(k)}(t) \triangleq \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$, *Ritz vectors*

12: *Let* $\mathbf{Q}^{(k)}(t)$ *consist of the first* r *columns of* $\mathbf{Q}_A^{(k)}(t)$.

13: ***end for***

14: ***end for***

We will later show that without enlarging the actual tracking subspace dimension r, the improvement resulted from multiple inner iterations is quite marginal so that the increased amount of computations is not worthwhile. As explained previously in Section 4.2.1 this is because tracking only r-dimensional subspace when the actual principal subspace dimension is $2r$ during the transition period leads to a poor data matrix approximation based on which the estimated subspace could only converge to the corresponding inaccurate principal subspace and never become close to the actual principal subspace. The consequence further strengthens the necessity of our previous statement to enlarge the actual tracking subspace dimension.

Due to the quadratic complexity of $O(NLr')$ resulted from matrix multiplication in Algorithm 4.3, more efficient subspace tracking algorithms are desired with lower complexity in practice. In the next two sections, we will provide several linear complexity algorithms based on Algorithm 4.3.

4.3 The First New Bi-Iteration SVD Subspace Tracking Algorithm, Bi-SVD1-MR

Recall the time varying data matrix $\mathbf{X}(t)$ using hybrid data window defined in (4.4). We will derive a linear complexity algorithm based on Algorithm 4.3 following the idea of sliding window adaptive SVD algorithms in [3] to reduce the complexity of $O(NLr')$ by using the original $\mathbf{X}(t)$. If we also apply multiple inner iterations, the low-rank consistent approximation $\hat{\mathbf{X}}^{(k)}(t)$ of $\mathbf{X}(t)$ at the kth inner iteration of time instant t could be expressed as

$$\hat{\mathbf{X}}^{(k)}(t) = \mathbf{Q}_B^{(k)}(t)\mathbf{A}^{(k)H}(t) = \mathbf{Q}_B^{(k)}(t)\mathbf{R}_A^{(k)H}(t)\mathbf{Q}_A^{(k)H}(t) \tag{4.7}$$

since it leaves $\mathbf{A}^{(k)}(t)$ in Algorithm 4.3 unaltered as

$$\mathbf{A}^{(k)}(t) = \mathbf{X}^H(t)\mathbf{Q}_B^{(k)}(t) = \hat{\mathbf{X}}^{(k)H}(t)\mathbf{Q}_B^{(k)}(t). \tag{4.8}$$

When a new data vector $\mathbf{x}(t)$ is available, the initialization data at time instant t are given by the results at the Kth inner iteration of time instant $t-1$, i.e. $\bar{\mathbf{Q}}_A^{(0)}(t) = \bar{\mathbf{Q}}_A^{(K)}(t-1)$, $\mathbf{U}^{(0)}(t) = \mathbf{U}^{(K)}(t-1)$, $\mathbf{R}_A^{(0)}(t) = \mathbf{R}_A^{(K)}(t-1)$, $\mathbf{Q}_B^{(0)}(t) = \mathbf{Q}_B^{(K)}(t-1)$, $\mathbf{R}_B^{(0)}(t) = \mathbf{R}_B^{(K)}(t-1)$. Thus, based on (4.7), $\mathbf{X}(t-1)$ could be approximated by

$$\begin{aligned}\hat{\mathbf{X}}^{(K)}(t-1) =& \mathbf{Q}_B^{(K)}(t-1)\mathbf{R}_A^{(K)H}(t-1)\mathbf{Q}_A^{(K)H}(t-1)\\ =& \mathbf{Q}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{Q}_A^{(0)H}(t).\end{aligned} \tag{4.9}$$

Therefore, at each time instant t we have

$$\hat{\mathbf{X}}(t) = \begin{bmatrix} \beta^{1/2}\mathbf{x}^H(t) \\ \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) \end{bmatrix} \tag{4.10}$$

where $\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) = [\mathbf{I}_{L-1}, \mathbf{0}]\hat{\mathbf{X}}^{(K)}(t-1)$, i.e. the first $L-1$ rows of $\hat{\mathbf{X}}_{L-1}^{(K)}(t-1)$. The above estimated $\hat{\mathbf{X}}(t)$ will be used to replace $\mathbf{X}(t)$ during the whole update procedure of $\mathbf{Q}_A^{(k)}(t)$ and other related data matrices to guarantee that the inner iterations operate on the same estimated data matrix. Thus, the columns of $\mathbf{Q}_A^{(k)}(t)$ actually converge to the r' principal right singular vectors of $\hat{\mathbf{X}}(t)$ using multiple inner iterations during each time constant t. As $\hat{\mathbf{X}}(t)$ becomes a more and more accurate estimate of $\mathbf{X}(t)$, the columns of $\mathbf{Q}_A^{(K)}(t)$ will essentially converge to the actual r' principal right singular vectors of $\mathbf{X}(t)$.

4.3.1 The First QR Decomposition

We start the algorithm derivation by updating the first QR decomposition in Algorithm 4.3. In terms of data matrix estimate in (4.10), the estimated $\mathbf{B}^{(k)}(t)$ can be given by

$$\hat{\mathbf{B}}^{(k)}(t) = \hat{\mathbf{X}}(t)\bar{\mathbf{Q}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t) = \begin{bmatrix} \beta^{1/2}\mathbf{x}^H(t) \\ \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) \end{bmatrix} \bar{\mathbf{Q}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t). \tag{4.11}$$

Let $\mathbf{h}^{(k-1)}(t) = \bar{\mathbf{Q}}_A^{(k-1)H}(t)\mathbf{x}(t)$ and $\mathbf{g}^{(k-1)}(t) = \mathbf{U}^{(k-1)H}(t)\mathbf{h}^{(k-1)}(t) = \mathbf{Q}_A^{(k-1)H}(t)\mathbf{x}(t)$. Then, we can have

$$\begin{aligned}
\hat{\mathbf{B}}^{(k)}(t) &= \begin{bmatrix} \beta^{1/2}\mathbf{g}^{(k-1)H}(t) \\ \alpha^{1/2}\begin{bmatrix}\mathbf{I}_{L-1} & \mathbf{0}\end{bmatrix}\hat{\mathbf{X}}^{(K)}(t-1)\bar{\mathbf{Q}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t)\end{bmatrix} \\
&= \begin{bmatrix} \beta^{1/2}\mathbf{g}^{(k-1)H}(t) \\ \alpha^{1/2}\begin{bmatrix}\mathbf{I}_{L-1} & \mathbf{0}\end{bmatrix}\mathbf{Q}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{Q}_A^{(0)H}(t)\bar{\mathbf{Q}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t)\end{bmatrix} \\
&= \begin{bmatrix} \beta^{1/2}\mathbf{g}^{(k-1)H}(t) \\ \alpha^{1/2}\begin{bmatrix}\mathbf{I}_{L-1} & \mathbf{0}\end{bmatrix}\mathbf{Q}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{U}^{(0)H}(t)\left(\bar{\mathbf{Q}}_A^{(0)H}(t)\bar{\mathbf{Q}}_A^{(k-1)}(t)\right)\mathbf{U}^{(k-1)}(t)\end{bmatrix} \\
&= \begin{bmatrix} \beta^{1/2}\mathbf{g}^{(k-1)H}(t) \\ \alpha^{1/2}\begin{bmatrix}\mathbf{I}_{L-1} & \mathbf{0}\end{bmatrix}\mathbf{Q}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t)\end{bmatrix} \\
&= \begin{bmatrix} \beta^{1/2}\mathbf{g}^{(k-1)H}(t) \\ \alpha^{1/2}\begin{bmatrix}\mathbf{I}_{L-1} & \mathbf{0}\end{bmatrix}\mathbf{Q}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t)\end{bmatrix}
\end{aligned} \tag{4.12}$$

where $\bar{\mathbf{\Theta}}_A^{(k-1)}(t) = \bar{\mathbf{Q}}_A^{(0)H}(t)\bar{\mathbf{Q}}_A^{(k-1)}(t) \in \mathbb{C}^{r'\times r'}$ and

$$\mathbf{\Theta}_A^{(k-1)}(t) = \mathbf{Q}_A^{(0)H}(t)\mathbf{Q}_A^{(k-1)}(t) = \mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k-1)}(t)\mathbf{U}^{(k-1)}(t) \in \mathbb{C}^{r'\times r'}. \tag{4.13}$$

Although $\bar{\mathbf{\Theta}}_A^{(0)}(t) = \mathbf{I}$, in general Nr'^2 flops and another $2r'^3$ flops are required in order to obtain $\mathbf{\Theta}_A^{(k-1)}(t)$ and $\bar{\mathbf{\Theta}}_A^{(k-1)}(t)$. It could be shown later that the r' by r' matrix $\bar{\mathbf{\Theta}}_A^{(k-1)}(t)$ could be obtained as a by-product of later updating scheme and the total computation complexity of $\mathbf{\Theta}_A^{(k-1)}(t)$ will be reduced to only $2r'^3$. Let $\mathbf{z} = [1, 0, \ldots, 0]^T$, $\mathbf{z}_L = [0, \ldots, 0, 1]^T$

and $\mathbf{q}_{B_L}(t) = \mathbf{Q}_B^{(0)H}(t)\mathbf{z}_L$ be the constant column vector given by conjugate transposing the last row of $\mathbf{Q}_B^{(0)}(t)$ at time instant t. Then, we can have

$$\begin{aligned}\hat{\mathbf{B}}^{(k)}(t) =& \alpha^{1/2} \begin{bmatrix} \mathbf{0}^T & 1 \\ \mathbf{I}_{L-1} & \mathbf{0} \end{bmatrix} \mathbf{Q}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) \\ &+ \mathbf{z}\left[\beta^{1/2}\mathbf{g}^{(k-1)H}(t) - \alpha^{1/2}\mathbf{q}_{B_L}^H(t)\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t)\right] \\ =& \alpha^{1/2}\tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{z}\tilde{\mathbf{g}}^{(k-1)H}(t) \end{aligned} \tag{4.14}$$

where $\tilde{\mathbf{g}}^{(k-1)}(t) = \beta^{1/2}\mathbf{g}^{(k-1)}(t) - \alpha^{1/2}\mathbf{\Theta}_A^{(k-1)H}(t)\mathbf{R}_A^{(0)}(t)\mathbf{q}_{B_L}(t)$, and

$$\tilde{\mathbf{Q}}_B^{(0)}(t) = \begin{bmatrix} \mathbf{0}^T & 1 \\ \mathbf{I}_{L-1} & \mathbf{0} \end{bmatrix} \mathbf{Q}_B^{(0)}(t). \tag{4.15}$$

It is easy to show that the columns of $\tilde{\mathbf{Q}}_B^{(0)}(t)$ are also orthonormal. Note that the computation of $\tilde{\mathbf{g}}^{(k-1)}(t)$ can be achieved by a series of matrix-vector product and has complexity of $2r'^2$.

Our goal here is to find a direct time updating scheme for the QR decomposition factors $\mathbf{Q}_B^{(k)}(t)$ and $\mathbf{R}_B^{(k)}(t)$ from $\hat{\mathbf{B}}^{(k)}(t)$. To achieve this, the constant vector $\mathbf{z}$ is decomposed into a component that lies in the "old" principal subspace spanned by the columns of $\mathbf{Q}_B^{(0)}(t)$ and an orthogonal innovation vector $\mathbf{z}_\perp(t)$. Based on the above definitions, we can have $\mathbf{q}_{B_L}(t) = \tilde{\mathbf{Q}}_B^{(0)H}(t)\mathbf{z} = \mathbf{Q}_B^{(0)H}(t)\mathbf{z}_L$. Thus, $\mathbf{z}_\perp(t)$ is given by

$$\mathbf{z}_\perp(t) = \left(\mathbf{I} - \tilde{\mathbf{Q}}_B^{(0)}(t)\tilde{\mathbf{Q}}_B^{(0)H}(t)\right)\mathbf{z} = \mathbf{z} - \tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{q}_{B_L}(t). \tag{4.16}$$

where $\tilde{\mathbf{Q}}_B^{(0)}(t)\tilde{\mathbf{Q}}_B^{(0)H}(t)$ is the orthogonal projector onto $R(\tilde{\mathbf{Q}}_B^{(0)}(t))$. By normalizing $\mathbf{z}_\perp(t)$, we can obtain

$$\mathbf{z} = \mu_z(t)\bar{\mathbf{z}}_\perp(t) + \tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{q}_{B_L}(t) \tag{4.17}$$

where $\mu_z(t) = (\mathbf{z}_\perp^H(t)\mathbf{z}_\perp(t))^{1/2}$ and $\bar{\mathbf{z}}_\perp(t) = \mathbf{z}_\perp(t)/\mu_z(t)$. Note that all the above data parameters $\mathbf{x}_\perp(t)$, $\mathbf{h}^{(1)}(t)$, $\mu_z(t)$ are not changing with different inner iteration index k and only needs to be calculated once for for fixed time instant index t. Then, replace $\mathbf{z}$ by (4.17) in (4.14) to obtain

$$\begin{aligned}
&\mathbf{Q}_B^{(k)}(t)\mathbf{R}_B^{(k)}(t) \\
=&\alpha^{1/2}\tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \left(\mu_z(t)\bar{\mathbf{z}}_\perp(t) + \tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{q}_{B_L}(t)\right)\tilde{\mathbf{g}}^{(k-1)H}(t) \\
=&\alpha^{1/2}\tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mu_z(t)\bar{\mathbf{z}}_\perp(t)\tilde{\mathbf{g}}^{(k-1)H}(t) + \tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{q}_{B_L}(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \\
=&\tilde{\mathbf{Q}}_B^{(0)}(t)\left(\alpha^{1/2}\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{q}_{B_L}(t)\tilde{\mathbf{g}}^{(k-1)H}(t)\right) + \mu_z(t)\bar{\mathbf{z}}_\perp(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \\
=&\begin{bmatrix}\tilde{\mathbf{Q}}_B^{(0)}(t) & \bar{\mathbf{z}}_\perp(t)\end{bmatrix}\begin{bmatrix}\alpha^{1/2}\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{q}_{B_L}(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \\ \mu_z(t)\tilde{\mathbf{g}}^{(k-1)H}(t)\end{bmatrix}.
\end{aligned} \tag{4.18}$$

Let $\mathbf{G}_B^{(k)}(t)$ be a sequence of $(r'^2/2 + r'/2)$ orthonormal Givens rotations that converts the second matrix in (4.18) to an upper triangular matrix $\mathbf{R}_B^{(k)}(t)$ with an additional zero row and we can have

$$\begin{bmatrix}\mathbf{Q}_B^{(k)}(t) & \mathbf{q}_{B\perp}^{(k)}(t)\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{Q}}_B^{(0)}(t) & \bar{\mathbf{z}}_\perp(t)\end{bmatrix}\mathbf{G}_B^{(k)H}(t) \tag{4.19}$$

$$\begin{bmatrix}\mathbf{R}_B^{(k)}(t) \\ 0\ldots 0\end{bmatrix} = \mathbf{G}_B^{(k)}(t)\begin{bmatrix}\alpha^{1/2}\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{q}_{B_L}(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \\ \mu_z(t)\tilde{\mathbf{g}}^{(k-1)H}(t)\end{bmatrix}. \tag{4.20}$$

Since 4 multiplication and one square root are needed to obtain each Givens rotation matrix, it requires about $2r'(r'+1)^2$ flops to explicitly calculate $\mathbf{G}_B^{(k)}(t)$. Then, the updated $\mathbf{Q}_B^{(k)}(t)$ could be obtained from (4.19) by rotating $[\tilde{\mathbf{Q}}_B^{(0)}(t), \bar{\mathbf{z}}_\perp(t)]$ using the sequence of Givens matrix $\mathbf{G}_B^{(k)H}(t)$ which requires about $2Lr'(r'+1)$ flops and then taking its first r' columns.

4.3.2 The Second QR Decomposition

In this part, we consider the second QR decomposition in Algorithm 4.3. Again, from (4.4) we can have

$$\begin{aligned}
\hat{\mathbf{A}}^{(k)}(t) =& \hat{\mathbf{X}}^H(t)\mathbf{Q}_B^{(k)}(t) \\
=& \begin{bmatrix} \beta^{1/2}\mathbf{x}(t) & \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)H}(t-1) \end{bmatrix} \mathbf{Q}_B^{(k)}(t) \\
=& \begin{bmatrix} \beta^{1/2}\mathbf{x}(t) & \alpha^{1/2}\hat{\mathbf{X}}^{(K)H}(t-1) \end{bmatrix} \begin{bmatrix} \mathbf{Q}_B^{(k)}(t) \\ 0 \dots 0 \end{bmatrix} \\
=& \begin{bmatrix} \beta^{1/2}\mathbf{x}(t) & \alpha^{1/2}\bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{U}^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{Q}_B^{(0)H}(t) \end{bmatrix} \begin{bmatrix} \mathbf{Q}_B^{(k)}(t) \\ 0 \dots 0 \end{bmatrix}
\end{aligned} \tag{4.21}$$

Similar to the operations on $\mathbf{z}$, we decompose the data vector $\mathbf{x}(t)$ into a component that lies in the "old" principal subspace spanned by the columns of $\bar{\mathbf{Q}}_A^{(0)}(t)$ and an orthogonal innovation vector $\mathbf{x}_\perp(t)$ and can obtain

$$\mathbf{x}_\perp(t) = \left(\mathbf{I} - \bar{\mathbf{Q}}_A^{(0)}(t)\bar{\mathbf{Q}}_A^{(0)H}(t)\right)\mathbf{x}(t) = \mathbf{x}(t) - \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t). \tag{4.22}$$

where $\bar{\mathbf{Q}}_A^{(0)}(t)\bar{\mathbf{Q}}_A^{(0)H}(t)$ is the orthogonal projector onto $R(\bar{\mathbf{Q}}_A^{(0)}(t))$. By normalizing $\mathbf{x}_\perp(t)$, we can obtain

$$\mathbf{x}(t) = \mu_x(t)\bar{\mathbf{x}}_\perp(t) + \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t) \tag{4.23}$$

where $\mu_x(t) = (\mathbf{x}_\perp^H(t)\mathbf{x}_\perp(t))^{1/2}$ and $\bar{\mathbf{x}}_\perp(t) = \mathbf{x}_\perp(t)/\mu_x(t)$. Note that all the above data parameters $\mathbf{x}_\perp(t)$, $\mathbf{h}^{(0)}(t)$, $\mu_x(t)$ are not changing with different inner iteration index k and only need to be computed once for each fixed time instant index t. Thus, $\hat{\mathbf{A}}^{(k)}(t)$ could be further approximated by

$$\begin{aligned}\hat{\mathbf{A}}^{(k)}(t) &= \begin{bmatrix}\beta^{1/2}\mu_x(t)\bar{\mathbf{x}}_\perp(t) + \beta^{1/2}\bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t) & \alpha^{1/2}\bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{U}^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{Q}_B^{(0)H}(t)\end{bmatrix}\begin{bmatrix}\mathbf{Q}_B^{(k)}(t)\\ 0\dots 0\end{bmatrix}\\ &= \begin{bmatrix}\bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t)\end{bmatrix}\begin{bmatrix}\beta^{1/2}\mathbf{h}^{(0)}(t) & \alpha^{1/2}\mathbf{U}^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{Q}_B^{(0)H}(t)\\ \beta^{1/2}\mu_x(t) & 0\dots 0\end{bmatrix}\begin{bmatrix}\mathbf{Q}_B^{(k)}(t)\\ 0\dots 0\end{bmatrix}.\end{aligned} \tag{4.24}$$

From Algorithm 4.3 we can also have

$$\begin{aligned}\bar{\mathbf{Q}}_A^{(0)H}(t)\hat{\mathbf{A}}^{(k)}(t) =& \bar{\mathbf{Q}}_A^{(0)H}(t)\hat{\mathbf{X}}^H(t)\mathbf{Q}_B^{(k)}(t)\\ =& \left(\mathbf{U}^{(0)}(t)\mathbf{U}^{(0)H}(t)\right)\bar{\mathbf{Q}}_A^{(0)H}(t)\hat{\mathbf{X}}^H(t)\mathbf{Q}_B^{(k)}(t)\\ =& \mathbf{U}^{(0)}(t)\left(\mathbf{Q}_A^{(0)H}(t)\hat{\mathbf{X}}^H(t)\right)\mathbf{Q}_B^{(k)}(t)\\ =& \mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{Q}_B^{(1)H}(t)\mathbf{Q}_B^{(k)}(t)\\ =& \mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t)\end{aligned} \tag{4.25}$$

where $\mathbf{\Theta}_B^{(k)}(t) = \mathbf{Q}_B^{(1)H}(t)\mathbf{Q}_B^{(k)}(t)$. By left multiplying both sides of (4.21) by $\bar{\mathbf{Q}}_A^{(0)H}(t)$, we

can obtain

$$\mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t) = \begin{bmatrix} \beta^{1/2}\mathbf{h}^{(0)}(t) & \alpha^{1/2}\mathbf{U}^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{Q}_B^{(0)H}(t) \end{bmatrix} \begin{bmatrix} \mathbf{Q}_B^{(k)}(t) \\ 0 \dots 0 \end{bmatrix}. \quad (4.26)$$

Then, replacing the term on the right side of (4.26) in (4.24) by its left side term, the following equation can be obtained

$$\hat{\mathbf{A}}^{(k)}(t) = \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t) \end{bmatrix} \begin{bmatrix} \mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t) \\ \beta^{1/2}\mu_x(t)\mathbf{q}_{B_1}^{(k)}(t) \end{bmatrix} \quad (4.27)$$

$$= \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{R}_A^{(k)}(t) \quad (4.28)$$

where $\mathbf{q}_{B_1}^{(k)}(t) = \mathbf{Q}_B^{(k)H}(t)\mathbf{z}$ is the column vector obtained by conjugate transposing the first row of $\mathbf{Q}_B^{(k)}(t)$ and

$$\begin{bmatrix} \bar{\mathbf{Q}}_A^{(k)}(t) & \mathbf{q}_{A\perp}^{(k)}(t) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t) \end{bmatrix} \mathbf{G}_A^{(k)H}(t) \quad (4.29)$$

$$\begin{bmatrix} \mathbf{R}_A^{(k)}(t) \\ 0 \dots 0 \end{bmatrix} = \mathbf{G}_A^{(k)}(t) \begin{bmatrix} \mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t) \\ \beta^{1/2}\mu_x(t)\mathbf{q}_{B_1}^{(k)}(t) \end{bmatrix}. \quad (4.30)$$

where similar to the first iteration $\mathbf{G}_A^{(k)}(t)$ is a sequence of $(r'^2/2 + r'/2)$ orthonormal Givens rotations that converts the second matrix in (4.27) to an upper triangular matrix $\mathbf{R}_A^{(k)}(t)$ with an additional zero row. It requires about $2r'(r'+1)^2$ and $2Nr'(r'+1)$ flops to compute $\mathbf{G}_A^{(k)}(t)$ and $\bar{\mathbf{Q}}_A^{(k)}(t)$. During the above update procedure, besides a sequence of Givens

matrix multiplication, another main computational burden results from the matrix product to compute $\mathbf{\Theta}_B^{(k)}(t)$ with Nr'^2 flops. To reduce the complexity, the update of (4.19) during the first iteration can be applied and we have

$$\begin{aligned}
&\begin{bmatrix} \mathbf{Q}_B^{(1)H}(t) \\ \mathbf{q}_{B\perp}^{(1)H}(t) \end{bmatrix} \begin{bmatrix} \mathbf{Q}_B^{(k)}(t) & \mathbf{q}_{B\perp}^{(k)}(t) \end{bmatrix} \\
=&\begin{bmatrix} \mathbf{\Theta}_B^{(k)}(t) & \mathbf{Q}_B^{(1)H}(t)\mathbf{q}_{B\perp}^{(k)}(t) \\ \mathbf{q}_{B\perp}^{(1)H}(t)\mathbf{Q}_B^{(k)}(t) & \mathbf{q}_{B\perp}^{(1)H}(t)\mathbf{q}_{B\perp}^{(k)}(t) \end{bmatrix} \\
=&\mathbf{G}_B^{(1)}(t) \begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)H}(t) \\ \bar{\mathbf{z}}_\perp^H(t) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)}(t) & \bar{\mathbf{z}}_\perp(t) \end{bmatrix} \mathbf{G}_B^{(k)H}(t) \\
=&\mathbf{G}_B^{(1)}(t)\mathbf{G}_B^{(k)H}(t).
\end{aligned} \tag{4.31}$$

Therefore, $\mathbf{\Theta}_B^{(k)}(t)$ can be directly extracted from the matrix product of $\mathbf{G}_B^{(1)}(t)\mathbf{G}_B^{(k)H}(t)$ with only $(r'+1)^3$ flops and $\mathbf{\Theta}_B^{(1)}(t) = \mathbf{I}_{r'+1}$.

From (4.29) we can also have

$$\begin{aligned}
\mathbf{G}_A^{(k)H}(t) &= \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)H}(t) \\ \bar{\mathbf{x}}_\perp^H(t) \end{bmatrix} \begin{bmatrix} \bar{\mathbf{Q}}_A^{(k)}(t) & \mathbf{q}_{A\perp}^{(k)}(t) \end{bmatrix} \\
&= \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)H}(t)\bar{\mathbf{Q}}_A^{(k)}(t) & \bar{\mathbf{Q}}_A^{(0)H}(t)\mathbf{q}_{A\perp}^{(k)}(t) \\ \bar{\mathbf{x}}_\perp^H(t)\bar{\mathbf{Q}}_A^{(k)}(t) & \bar{\mathbf{x}}_\perp^H(t)\mathbf{q}_{A\perp}^{(k)}(t) \end{bmatrix}
\end{aligned} \tag{4.32}$$

Thus, $\bar{\mathbf{\Theta}}_A^{(k)}(t) = \bar{\mathbf{Q}}_A^{(0)H}(t)\bar{\mathbf{Q}}_A^{(k)}(t)$ can be directly extracted from $\mathbf{G}_A^{(k)}(t)$ as a by-product as mentioned earlier and involved in the computation of the first QR decomposition at the

$(k+1)$th inner iteration.

If we let $\bar{\mathbf{Q}}_A^{(k)}(t)$ replaced by $\mathbf{Q}_A^{(k)}(t)$ and $\mathbf{U}^{(k)}(t)$ be constantly an identity matrix during each time instant t, so far the extension of bi-iteration SVD algorithm to multiple inner iterations but without Ritz acceleration is already described for Algorithm 4.3, called Bi-SVD1-M-1.

4.3.3 Ritz Acceleration

In this part we integrate the idea of Ritz acceleration into bi-iteration SVD algorithm for subspace tracking to improve the algorithm performance. Since the SVD in Step 9 and 10 of Algorithm 4.3 deals with a larger dimension $\hat{\mathbf{T}}^{(k)}(t)$, the ED of a r' by r' matrix is used to implement Ritz acceleration. Again, the estimate $\hat{\mathbf{X}}(t)$ replaces $\mathbf{X}(t)$ in Step 9 of Algorithm 4.3 and from (4.10) we can obtain that

$$
\begin{aligned}
&\hat{\mathbf{X}}^H(t)\hat{\mathbf{X}}(t) \\
=&\begin{bmatrix} \beta^{1/2}\mathbf{x}(t) & \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)H}(t-1) \end{bmatrix} \begin{bmatrix} \beta^{1/2}\mathbf{x}^H(t) \\ \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) \end{bmatrix} \\
=&\alpha\hat{\mathbf{X}}_{L-1}^{(K)H}(t-1)\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) + \beta\mathbf{x}(t)\mathbf{x}^H(t) \\
=&\alpha\hat{\mathbf{X}}^{(K)H}(t-1)\hat{\mathbf{X}}^{(K)}(t-1) - \alpha\mathbf{p}(t)\mathbf{p}^H(t) + \beta\mathbf{x}(t)\mathbf{x}^H(t) \\
=&\alpha\mathbf{Q}_A^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{Q}_A^{(0)H}(t) - \alpha\mathbf{p}(t)\mathbf{p}^H(t) + \beta\mathbf{x}(t)\mathbf{x}^H(t)
\end{aligned} \tag{4.33}
$$

where $\mathbf{p}(t)$ is the conjugate transpose of the last row of $\hat{\mathbf{X}}^{(K)}(t-1)$ which can be given by

$$\mathbf{p}(t) = \hat{\mathbf{X}}^{(K)H}(t-1)\mathbf{z}_L = \mathbf{Q}_A^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{Q}_B^{(0)H}(t)\mathbf{z}_L = \mathbf{Q}_A^{(0)}(t)\mathbf{s}(t), \tag{4.34}$$

and $\mathbf{s}(t) = \mathbf{R}_A^{(0)}(t)\mathbf{q}_{B_L}(t)$. Thus, $\hat{\mathbf{T}}^{(k)}(t)$ can be given by

$$\begin{aligned}
\hat{\mathbf{T}}^{(k)}(t) =& \alpha\bar{\mathbf{Q}}_A^{(k)H}(t)\mathbf{Q}_A^{(0)}(t)\mathbf{R}_A^{(0)}(t)\mathbf{R}_A^{(0)H}(t)\mathbf{Q}_A^{(0)H}(t)\bar{\mathbf{Q}}_A^{(k)}(t) \\
& - \alpha\bar{\mathbf{Q}}_A^{(k)H}(t)\mathbf{Q}_A^{(0)}(t)\mathbf{s}(t)\mathbf{s}^H(t)\mathbf{Q}_A^{(0)H}(t)\mathbf{Q}_A^{(0)H}(t) + \beta\mathbf{h}^{(k)}(t)\mathbf{h}^{(k)H}(t) \\
=& \alpha\left(\bar{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{U}^{(0)}(t)\mathbf{R}_A^{(0)}(t)\right)\left(\mathbf{R}_A^{(0)H}(t)\mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k)}(t)\right) \\
& - \alpha\left(\bar{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{U}^{(0)}(t)\mathbf{s}(t)\right)\left(\mathbf{s}^H(t)\mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k)}(t)\right) + \beta\mathbf{h}^{(k)}(t)\mathbf{h}^{(k)H}(t) \\
=& \alpha\left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{R}_A^{(0)}(t)\right)\left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{R}_A^{(0)}(t)\right)^H \\
& - \alpha\left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{s}(t)\right)\left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{s}(t)\right)^H + \beta\mathbf{h}^{(k)}(t)\mathbf{h}^{(k)H}(t)
\end{aligned} \tag{4.35}$$

where

$$\mathbf{h}^{(k)}(t) = \bar{\mathbf{Q}}_A^{(k)H}(t)\mathbf{x}(t) \tag{4.36}$$

could be used in the first QR decomposition at the $(k+1)$th inner iteration and

$$\tilde{\mathbf{\Theta}}_A^{(k)}(t) = \mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k)}(t) \tag{4.37}$$

could also be used to compute $\bar{\mathbf{\Theta}}_A^{(k)}(t)$ later in the first iteration of the $(k+1)$th inner iteration so that $\mathbf{\Theta}_A^{(k)}(t) = \tilde{\mathbf{\Theta}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$ in (4.13). Thus, in order to obtain $\hat{\mathbf{T}}^{(k)}(t)$, a total of $3r'^3 + Nr' + 3r'^2$ flops are required and $r'^3 + Nr'$ of them could be used later,

given $\mathbf{U}^{(0)}(t)$, $\bar{\mathbf{\Theta}}_A^{(k)}(t)$, $\mathbf{R}_A^{(0)}(t)$, $\mathbf{q}_{B_L}(t)$, $\bar{\mathbf{Q}}_A^{(k)}(t)$ and $\mathbf{x}(t)$. In fact, when the number of inner iterations K is large enough, $\mathbf{U}^{(0)}(t) = \mathbf{U}^{(K)}(t-1)$ will essentially become an identity matrix so that $\tilde{\mathbf{\Theta}}_A^{(k)}(t) \approx \bar{\mathbf{\Theta}}_A^{(k)}(t)$ and another r'^3 flops could be saved. In simulation we could see that about 3 to 5 inner iterations will be enough for that as the convergence speed of $\bar{\mathbf{Q}}_A^{(k)}(t)$ becomes very fast with Ritz acceleration. Thus, such a small number of inner iterations will not impose much computational burden on the proposed algorithm to achieve great performance improvement.

Then, the ED of $\hat{\mathbf{T}}^{(k)}(t)$ can be given by

$$\hat{\mathbf{T}}^{(k)}(t) = \mathbf{U}^{(k)}(t)\mathbf{D}^{(k)}(t)\mathbf{U}^{(k)H}(t). \tag{4.38}$$

which requires $O(r'^3)$ flops and can be obtained very efficiently by symmetric QR algorithm. Thus, $\mathbf{Q}^{(k)}(t)$ can be expressed by the first r columns of $\mathbf{Q}_A^{(k)}(t) \triangleq \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$. Moreover, to avoid the matrix product of Nr'^2 flops, $\mathbf{Q}_A^{(k)}(t)$ does not have to be explicitly computed as in the previous derivations it is always substituted by $\bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$. Since $\mathbf{U}^{(k)}(t)$ only plays the role of a rotation matrix to accelerate the convergence process, the estimated r principal eigenvectors included in $\mathbf{Q}^{(K)}(t)$ at the last inner iterations could also be estimated as the first r columns of $\bar{\mathbf{Q}}_A^{(K)}(t)$ without the additional rotation imposed by $\mathbf{U}^{(k)}(t)$. In fact, during one data update the impact of Ritz acceleration by $\mathbf{U}^{(k)}(t)$ has already been made on $\bar{\mathbf{Q}}_A^{(k)}(t)$ at the k inner iterations, $k = 1, \ldots, K-1$. Since we only need one explicit $\mathbf{Q}^{(K)}(t)$ at the last inner iteration for each data update for further applications, it can always be replaced by the first r columns of $\bar{\mathbf{Q}}_A^{(K)}(t)$ but still with accelerated convergence

rate.

Again, when we have a large enough K (only about 3 to 5), $\mathbf{Q}^{(K)}(t)$ will be very close to an identity matrix so that $\bar{\mathbf{Q}}_A^{(k)}(t)$ could be directly used as $\mathbf{Q}_A^{(k)}(t)$ with little performance penalty, which reduces the algorithm complexity by a large amount of flops Nr'^2. Together with the above procedure to compute $\hat{\mathbf{T}}^{(k)}(t)$, we emphasize the impact of Ritz acceleration on the overall algorithm complexity: while it introduces more computational loads by extra matrix product and eigendecomposition, parts of the loads could actually be eliminated by the impressive benefit of the accelerated convergence rate as $\mathbf{U}^{(k)}(t)$ approaches the identity matrix very fast after only a few inner iterations and the number of inner iterations could be reduced to achieve the same performance compared to algorithms without Ritz acceleration.

4.3.4 Algorithm Description, Bi-SVD1-MR-1

The overall low complexity version of Algorithm 4.3 with Ritz acceleration and multiple inner iterations with complexity $O(Nr'^2) + O(Lr'^2)$ for subspace tracking is described as in Algorithm 4.4 as Bi-SVD1-MR-1.

Algorithm 4.4 (Bi-SVD1-MR-1)**.**

1: ***Initialization:*** $\bar{\mathbf{Q}}_A^{(K)}(0) = [\mathbf{I}_{r'}, \mathbf{0}]^T$, $\mathbf{U}^{(K)}(0) = \mathbf{I}_{r'}$, $\mathbf{R}_A^{(K)}(0) = \mathbf{I}_{r'}$, *and* $\mathbf{Q}_B^{(K)}(0) = [\mathbf{I}_{r'}, \mathbf{0}]^T$. *Let* $\mathbf{z} = [1, 0, \ldots, 0]^T$, $\mathbf{z}_L = [0, \ldots, 0, 1]^T$.

2: ***for*** $t = 1, 2, \ldots$ ***do***

3: ***Initialization:*** *Let* $\bar{\mathbf{Q}}_A^{(0)}(t) = \bar{\mathbf{Q}}_A^{(K)}(t-1)$, $\mathbf{U}^{(0)}(t) = \mathbf{U}^{(K)}(t-1)$, $\bar{\mathbf{\Theta}}_A^{(0)}(t) = \mathbf{I}_{r'}$, $\mathbf{R}_A^{(0)}(t) = \mathbf{R}_A^{(K)}(t-1)$, *and* $\mathbf{Q}_B^{(0)}(t) = \mathbf{Q}_B^{(K)}(t-1)$.

4: $\mathbf{h}^{(0)}(t) = \bar{\mathbf{Q}}_A^{(0)H}(t)\mathbf{x}(t)$, $\mathbf{q}_{B_L}(t) = \mathbf{Q}_B^{(0)H}(t)\mathbf{z}_L$, $\mathbf{s}(t) = \mathbf{R}_A^{(0)}(t)\mathbf{q}_{B_L}(t)$ —$Nr' + r'^2$

5: $$\tilde{\mathbf{Q}}_B^{(0)}(t) = \begin{bmatrix} \mathbf{0}^H & 1 \\ \mathbf{I}_{L-1} & \mathbf{0} \end{bmatrix} \mathbf{Q}_B^{(0)}(t)$$

6: $\mathbf{z}_\perp(t) = \mathbf{z} - \tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{q}_{B_L}(t)$, $\mu_z(t) = (\mathbf{z}_\perp^H(t)\mathbf{z}_\perp(t))^{1/2}$, $\bar{\mathbf{z}}_\perp(t) = \mathbf{z}_\perp(t)/\mu_z(t)$ —Nr'

7: $\mathbf{x}_\perp(t) = \mathbf{x}(t) - \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t)$, $\mu_x(t) = (\mathbf{x}_\perp^H(t)\mathbf{x}_\perp(t))^{1/2}$, $\bar{\mathbf{x}}_\perp(t) = \mathbf{x}_\perp(t)/\mu_x(t)$ —Nr'

8: ***for*** $k = 1, 2, \ldots, K$ ***do***

9: ***First iteration:***

10: $\mathbf{g}^{(k-1)}(t) = \mathbf{U}^{(k-1)H}(t)\mathbf{h}^{(k-1)}(t)$ —Nr'

11: $\tilde{\mathbf{g}}^{(k-1)}(t) = \beta^{1/2}\mathbf{g}^{(k-1)}(t) - \alpha^{1/2}\mathbf{\Theta}_A^{(k-1)H}(t)\mathbf{R}_A^{(0)}(t)\mathbf{q}_{B_L}(t)$ —$2r'^2$

12: $$\begin{bmatrix} \mathbf{R}_B^{(k)}(t) \\ 0 \ldots 0 \end{bmatrix} = \mathbf{G}_B^{(k)}(t) \begin{bmatrix} \alpha^{1/2}\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{q}_{B_L}(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \\ \mu_z(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \end{bmatrix}$$ — $r'^3 + r'^2 + 2r'(r'+1)^2$

13: ***if*** $k = K$ ***then***

14: $\begin{bmatrix} \mathbf{Q}_B^{(k)}(t) & \mathbf{q}_{B\perp}^{(k)}(t) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)}(t) & \bar{\mathbf{z}}_\perp(t) \end{bmatrix} \mathbf{G}_B^{(k)H}(t)$ —*about* $2Lr'^2$ *only if* $k = K$

15: ***end if***

16: *Extract* $\mathbf{\Theta}_B^{(k)}(t)$ *from* $\mathbf{G}_B^{(1)}(t)\mathbf{G}_B^{(k)H}(t)$ *by* (4.31)

17: ***Second iteration:***

18: $$\mathbf{q}_{B_1}^{(k)}(t) = \bar{\mathbf{G}}_B^{(k)}(t) \left(\begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)H}(t) \\ \bar{\mathbf{z}}_\perp^H(t) \end{bmatrix} \mathbf{z} \right)$$ —$r'^2 + r'$

19: $$\begin{bmatrix} \mathbf{R}_A^{(k)}(t) \\ 0 \ldots 0 \end{bmatrix} = \mathbf{G}_A^{(k)}(t) \begin{bmatrix} \mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t) \\ \beta^{1/2}\mu_x(t)\mathbf{q}_{B_1}^{(k)}(t) \end{bmatrix}$$ —$2r'^3 + 2r'(r'+1)^2$

20: ***if*** $k = K$ ***then***

21: $\begin{bmatrix} \bar{\mathbf{Q}}_A^{(k)}(t) & \mathbf{q}_{A\perp}^{(k)}(t) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t) \end{bmatrix} \mathbf{G}_A^{(k)H}(t)$ —*about* $2Nr'^2$ *only if* $k = K$

22: ***end if***

23: *Extract* $\bar{\mathbf{\Theta}}_A^{(k)}(t)$ *from* $\mathbf{G}_A^{(k)}(t)$ *by* (4.32)

24: ***Ritz Acceleration****:*

25: $$\mathbf{h}^{(k)}(t) = \bar{\mathbf{G}}_A^{(k)}(t) \begin{bmatrix} \mathbf{h}^{(0)}(t) \\ \bar{\mathbf{x}}_\perp^H(t)\mathbf{x}(t) \end{bmatrix} \quad —r'^2 + r'$$

26: $$\tilde{\mathbf{\Theta}}_A^{(k)}(t) = \mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k)}(t) \quad —r'^3$$

27: $$\mathbf{S}^{(k)}(t) = \tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{R}_A^{(0)}(t),\ \mathbf{w}^{(k)}(t) = \tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{s}(t) \quad —r'^3 + 2r'^2$$

28: $$\hat{\mathbf{T}}^{(k)}(t) = \alpha\mathbf{S}^{(k)}(t)\mathbf{S}^{(k)H}(t) - \alpha\mathbf{w}^{(k)}(t)\mathbf{w}^{(k)H}(t) + \beta\mathbf{h}^{(k)}(t)\mathbf{h}^{(k)H}(t) \quad —r'^3 + r'^2$$

29: $$\hat{\mathbf{T}}^{(k)}(t) = \mathbf{U}^{(k)}(t)\mathbf{D}^{(k)}(t)\mathbf{U}^{(k)H}(t),\ ED \quad —O(r'^3)$$

30: ***end for***

31: *Let* $\mathbf{Q}^{(k)}(t)$ *be the first* r *columns of* $\mathbf{Q}_A^{(k)}(t) \triangleq \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$ *or* $\bar{\mathbf{Q}}_A^{(k)}(t)$ —Nr'^2 *or zero*

32: ***end for***

Compared to previous descriptions, some modifications are made to further reduce algorithm complexity. First, we can see that during the whole algorithm derivation $\mathbf{Q}_B^{(k)}(t)$, $k = 1, \ldots, K$, is only used once for computing $\mathbf{q}_{B_1}^{(k)}(t)$ in (4.27) except when transferred to the next time instant $t+1$ for initialization as $\mathbf{Q}_B^{(K)}(t)$. Then, based on (4.19), $\mathbf{q}_{B_1}^{(k)}(t)$ could be obtained by

$$\mathbf{q}_{B_1}^{(k)}(t) = \bar{\mathbf{G}}_B^{(k)}(t) \left(\begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)H}(t) \\ \bar{\mathbf{z}}_\perp^H(t) \end{bmatrix} \mathbf{z} \right) \tag{4.39}$$

by $r'^2 + r'$ flops at each inner iteration, where $\bar{\mathbf{G}}_B^{(k)}(t) = [\mathbf{I}_{r'}, \mathbf{0}]\mathbf{G}_B^{(k)}(t)$ consists of the first

r' rows of $\mathbf{G}_B^{(k)}(t)$. Besides, only about $2Lr'(r'+1)$ flops to compute $\mathbf{Q}_B^{(K)}(t)$ in (4.19) are required at the Kth inner iteration as in Step 14 no matter how many inner iterations are assigned for each data update.

When it comes to $\mathbf{Q}_A^{(k)}(t)$, $k = 1, \ldots, K$, it only appears in (4.36) in the second QR decomposition during each data update other than being transferred to $(t+1)$ for initialization as $\mathbf{Q}_A^{(K)}(t)$. In fact, $\mathbf{Q}_A^{(k)}(t)$ is not necessary to be explicitly computed for (4.36). According to (4.29), $\mathbf{h}^{(k)}(t)$ in (4.36) could be rewritten as

$$\begin{aligned}\mathbf{h}^{(k)}(t) =& \bar{\mathbf{Q}}_A^{(k)H}(t)\mathbf{x}(t) \\ =& \bar{\mathbf{G}}_A^{(k)}(t) \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t) \end{bmatrix}^H \mathbf{x}(t) \\ =& \bar{\mathbf{G}}_A^{(k)}(t) \begin{bmatrix} \mathbf{h}^{(0)}(t) \\ \bar{\mathbf{x}}_\perp^H(t)\mathbf{x}(t) \end{bmatrix}\end{aligned} \tag{4.40}$$

where $\bar{\mathbf{G}}_A^{(k)}(t) = [\mathbf{I}_{r'}, \mathbf{0}]\mathbf{G}_A^{(k)}(t)$ consists of the first r' rows of $\mathbf{G}_A^{(k)}(t)$. Meanwhile, the computations required to compute $\mathbf{h}^{(k)}(t)$ are reduced from Nr' to $r'^2 + r'$. As the same as $\mathbf{Q}_B^{(K)}(t)$, fortunately, although the $2Nr'(r'+1)$ flops to compute $\mathbf{Q}_A^{(K)}(t)$ as the initialization transferred to $t+1$ in (4.29) are inevitable, they only need to be consumed once at the Kth inner iteration no matter how large K is. So far, all the steps in Bi-SVD1-MR-1 have linear complexity of $O(Nr') + O(Lr')$ or less $O(r'^3)$ when $N \gg r'$ so that $N > r'^2$, except Step 14 and 21 for about $2Nr'^2 + 2Lr'^2$ but only happen once for each data update.

Furthermore, when K is large enough as mentioned earlier, $\mathbf{U}^{(0)}(t)$ will converge to an identity matrix and $\mathbf{R}_A^{(0)}(t)$, $\mathbf{R}_B^{(0)}(t)$ will converge to the diagonal matrix including

the r' principal singular values. Thus, all the $\mathbf{U}^{(0)}(t)$ could be replaced by $\mathbf{I}_r$ and removed from the proposed algorithms so that another $2r'^3$ could be further reduced from Step 19 and 26.

4.3.5 Ultrafast Version, Bi-SVD1-MR-2

Even if $r' \ll N$, when N is large, r' could also be a large dimension to track. Under this condition, the steps with $O(Nr'^2)$ or $O(Lr'^2)$ could consume more computations than all the other steps even with multiple inner iterations and become the bottleneck of the algorithm complexity. In case that faster tracking algorithms are desired, at the Kth inner iteration we could take the upper Hessenberg part, i.e. upper triangular part plus a number of $l < r'$ subdiagonals below the main diagonal, of one component in (4.20) and (4.30) so that only a sequence of about $(l+2)r'$ Givens rotations for (4.20) or about $(l+2)r'$ Givens rotations for (4.30) is required to explicitly compute $\mathbf{Q}_B^{(K)}(t)$ and $\mathbf{Q}_A^{(K)}(t)$. Specifically, for (4.20) we replace $\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t)$ by its upper Hessenberg part. Then, the second matrix on the right hand side of (4.20) could be rewritten as the sum of an upper Hessenberg plus one additional row with all zeros and a rank one matrix by

$$\begin{bmatrix} Hess\left(\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t)\right) \\ \mathbf{0}^T \end{bmatrix} + \begin{bmatrix} \mathbf{q}_{B_L}(t) \\ \mu_z(t) \end{bmatrix} \tilde{\mathbf{g}}^{(k-1)H}(t). \tag{4.41}$$

We will see that although $\mathbf{\Theta}_A^{(k-1)}(t)$ is not an identity matrix, most of its weights are concentrated on the main diagonal and its neighboring subdiagonals. Moreover, when K is large enough, $\mathbf{R}_A^{(0)}(t)$ is almost diagonal with diagonal elements in descending order so

the elements in lower rows of $\mathbf{\Theta}_A^{(k-1)}(t)$ will be suppressed. Thus, taking the Hessenberg part of $\mathbf{R}_A^{(0)H}(t)\mathbf{\Theta}_A^{(k-1)}(t)$ keeps its main weights either around its main diagonal or in the upper rows. To rotate (4.41) into a upper triangular matrix, we could first use $r' - l - 1$ Givens rotations to convert $[\mathbf{q}_{B_L}^T(t), \mu_z(t)]^T$ to a vector with all zeros at the last $r' - l - 1$ rows. Then, the corresponding upper Hessenberg plus one addition all zero row will become an upper Hessenberg with one more subdiagonal and could be convert to upper triangular matrix by about $(l+1)r'$ Givens rotations. Similarly, for $\mathbf{Q}_A^{(K)}(t)$ in (4.30), we take the upper Hessenburg part of $\mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t)$, where $\mathbf{U}^{(0)}(t)$ is considered as identity matrix, and the second matrix on the right hand side of (4.30) could be rewritten as the sum of an upper Hessenberg plus one additional row with all zeros and a rank one matrix. Similar procedure could be operated to convert it to an upper triangular matrix. However, in this case $\mathbf{R}_B^{(1)H}(t)$ is a lower triangular matrix instead of $\mathbf{R}_A^{(0)}(t)$ that could be considered as diagonal. As a consequence, the lower triangular part of $\mathbf{U}^{(0)}(t)\mathbf{R}_B^{(1)H}(t)\mathbf{\Theta}_B^{(k)}(t)$ is not trivial any more so taking its upper Hessenburg part may yield an approximation matrix with substantial distortion. Furthermore, this faster version algorithm could fail to produce excellent tracking performance. This motivates the algorithms proposed in the next section, Bi-SVD2-MR, where only near diagonal matrices are involved in extracting upper Hessenburg part for QR decomposition.

In order to keep the near diagonal property of $\mathbf{R}_A^{(0)}(t)$ for better approximation by taking the upper Hessenburg part of certain matrix component, its initializations at each time instant will be assigned as $\mathbf{R}_A^{(K-1)}(t-1)$ at the $(K-1)$th inner iteration of the previous time instant, i.e. at time instant t, $\mathbf{R}_A^{(0)}(t) = \mathbf{R}_A^{(K-1)}(t-1)$. Therefore, the large

amount of computations involved in Step 14 of Bi-SVD1-MR-1 could be further reduced and we call this even lower complexity algorithm, Bi-SVD1-MR-2. Correspondingly, the same simplifications could also be applied to Bi-SVD1-M-1 and generate its faster version, Bi-SVD1-M-2.

4.4 The Second New Bi-Iteration SVD Subspace Tracking Algorithm, Bi-SVD2-MR

Recall the time varying data matrix $\mathbf{X}(t)$ using hybrid data window defined in (4.4). We will derive a linear complexity algorithm based on Algorithm 4.3 following the idea of approximating $\mathbf{X}(t)$ by a lower-rank matrix in [58]. When adapted to multiple inner iterations, the low-rank consistent approximation matrix $\hat{\mathbf{X}}^{(k)}(t)$ of $\mathbf{X}(t)$ at the kth inner iteration for time instant t is given by

$$\hat{\mathbf{X}}^{(k)}(t) = \mathbf{B}^{(k)}(t)\mathbf{Q}_A^{(k-1)H}(t) = \mathbf{Q}_B^{(k)}(t)\mathbf{R}_B^{(k)}(t)\mathbf{Q}_A^{(k-1)H}(t) \tag{4.42}$$

since it leaves $\mathbf{B}^{(k)}(t)$ in Algorithm 4.3 unaltered as

$$\mathbf{B}^{(k)}(t) = \mathbf{X}(t)\mathbf{Q}_A^{(k-1)}(t) = \hat{\mathbf{X}}^{(k)}(t)\mathbf{Q}_A^{(k-1)}(t). \tag{4.43}$$

When a new data vector $\mathbf{x}(t)$ is available, the initialization data at time instant t are provided by the tracking results at the $(K-1)$th and Kth inner iterations of time instant $t-1$, where $K \geq 2$ is the total number of inner iteration at each time instant as defined

before. Let $\mathbf{Q}_A^{(-1)H}(t) = \mathbf{Q}_A^{(K-1)H}(t-1)$, i.e. $\bar{\mathbf{Q}}_A^{(-1)}(t) = \bar{\mathbf{Q}}_A^{(K-1)}(t-1)$ and $\mathbf{U}^{(-1)}(t) = \mathbf{U}^{(K-1)}(t-1)$, $\mathbf{R}_A^{(0)}(t) = \mathbf{R}_A^{(K)}(t-1)$, $\mathbf{Q}_B^{(0)}(t) = \mathbf{Q}_B^{(K)}(t-1)$, $\mathbf{R}_B^{(0)}(t) = \mathbf{R}_B^{(K)}(t-1)$. Then, according to (4.42) $\mathbf{X}(t-1)$ could be approximated by

$$\begin{aligned}\hat{\mathbf{X}}^{(K)}(t-1) =& \mathbf{Q}_B^{(K)}(t-1)\mathbf{R}_B^{(K)}(t-1)\mathbf{Q}_A^{(K-1)H}(t-1) \\ =& \mathbf{Q}_B^{(0)}(t)\mathbf{R}_B^{(0)}(t)\mathbf{Q}_A^{(-1)H}(t). \end{aligned} \tag{4.44}$$

Therefore, according to (4.4) at each time instant t we have

$$\hat{\mathbf{X}}(t) = \begin{bmatrix} \beta^{1/2}\mathbf{x}^H(t) \\ \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) \end{bmatrix} \tag{4.45}$$

where $\hat{\mathbf{X}}_{L-1}^{(K)}(t-1) = [\mathbf{I}_{L-1}, \mathbf{0}]\hat{\mathbf{X}}^{(K)}(t-1)$, i.e. the first $L-1$ rows of $\hat{\mathbf{X}}_{L-1}^{(K)}(t-1)$. Note that the above estimated $\hat{\mathbf{X}}(t)$ will be used to replace $\mathbf{X}(t)$ during the whole update procedure of $\mathbf{Q}_A^{(k)}(t)$ and other related data matrices so that the inner iterations operate on the same estimated data matrix. Thus, the columns of $\mathbf{Q}_A^{(k)}(t)$ actually converge to the r' principal right singular vectors of the estimated data matrix $\hat{\mathbf{X}}(t)$ instead of $\mathbf{X}(t)$ using multiple inner iterations for each time instant t. As $\hat{\mathbf{X}}(t)$ becomes a more and more accurate estimate of $\mathbf{X}(t)$, the columns of $\mathbf{Q}_A^{(K)}(t)$ will essentially converge to the actual r' principal right singular vectors of $\mathbf{X}(t)$.

At time instant $(t-1)$, as k increases the resulting $\mathbf{Q}_A^{(k)}(t-1)$ will tend to converge so that $\mathbf{Q}_A^{(k-1)}(t-1)$ will be equivalent to $\mathbf{Q}_A^{(k)}(t-1)$ when k is large enough. In that case, we can also replace $\mathbf{Q}_A^{(-1)}(t) = \mathbf{Q}_A^{(K-1)}(t-1)$ by $\mathbf{Q}_A^{(0)}(t) = \mathbf{Q}_A^{(K)}(t-1)$, i.e. $\bar{\mathbf{Q}}_A^{(0)}(t) = \bar{\mathbf{Q}}_A^{(K)}(t-1)$

and $\mathbf{U}^{(0)}(t) = \mathbf{U}^{(K)}(t-1)$, in (4.44) and correspondingly let

$$\hat{\mathbf{X}}^{(K)}(t-1) = \mathbf{Q}_B^{(0)}(t)\mathbf{R}_B^{(0)}(t)\mathbf{Q}_A^{(0)H}(t). \tag{4.46}$$

Then, the columns of $\mathbf{Q}_A^{(k)}(t)$ actually converge to the r' principal right singular vectors of $\hat{\mathbf{X}}(t)$ in (4.45) with $\hat{\mathbf{X}}^{(K)}(t-1)$ replaced by (4.46) instead of (4.44). As long as K is large enough, the above data matrix approximation would lead to little performance degradation.

Since our main purpose is to investigate the performance of multiple inner iterations along with Ritz acceleration, the approximation of (4.46) for $\hat{\mathbf{X}}^{(K)}(t-1)$ will be employed assuming K is large enough. It turns out that $K = 3-5$ is large enough for algorithms with Ritz to yield excellent performance as it greatly accelerates the convergence rate. For smaller K, it is also easy to derive the corresponding algorithms in terms of the following procedure through substituting $\hat{\mathbf{X}}^{(K)}(t-1)$ with (4.44).

4.4.1 The First QR Decomposition

In this part we update the first QR decomposition in Algorithm 4.3 with lower complexity. The derivations are almost the same as what we did in Section 4.3.1 only with $\mathbf{R}_A^{(k)H}(t)$ replaced by $\mathbf{R}_B^{(k)}(t)$. Therefore, we omit the specific derivations and directly describe the results here. Again, let $\mathbf{G}_B^{(k)}(t)$ be a sequence of $(r'^2/2 + r'/2)$ orthonormal Givens rotations that converts the second matrix in (4.48) to an upper triangular matrix $\mathbf{R}_B^{(k)}(t)$ with an additional zero row and we can have

$$\begin{bmatrix} \mathbf{Q}_B^{(k)}(t) & \mathbf{q}_{B\perp}^{(k)}(t) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)}(t) & \bar{\mathbf{z}}_\perp(t) \end{bmatrix} \mathbf{G}_B^{(k)H}(t) \tag{4.47}$$

$$\begin{bmatrix} \mathbf{R}_B^{(k)}(t) \\ 0 \dots 0 \end{bmatrix} = \mathbf{G}_B^{(k)}(t) \begin{bmatrix} \alpha^{1/2} \mathbf{R}_B^{(0)}(t) \mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{q}_{B_L}(t) \tilde{\mathbf{g}}^{(k-1)H}(t) \\ \mu_z(t) \tilde{\mathbf{g}}^{(k-1)H}(t) \end{bmatrix}. \tag{4.48}$$

Since 4 multiplication and one square root are needed to obtain each Givens rotation matrix, we assume that it requires about $2r'(r'+1)^2$ flops to explicitly calculate $\mathbf{G}_B^{(k)}(t)$. The updated $\mathbf{Q}_B^{(k)}(t)$ could be obtained from (4.47) by rotating $[\tilde{\mathbf{Q}}_B^{(0)}(t), \bar{\mathbf{z}}_\perp(t)]$ using the sequence of Givens matrix $\mathbf{G}_B^{(k)H}(t)$ which requires about $2Lr'(r'+1)$ flops and then taking its first r' columns.

4.4.2 The Second QR Decomposition

Now we consider the second QR decomposition in Algorithm 4.3. Again, from (4.4) we can have

$$
\begin{aligned}
&\mathbf{A}^{(k)}(t)\mathbf{R}_B^{(k)}(t) \\
=&\hat{\mathbf{X}}^H(t)\mathbf{Q}_B^{(k)}(t)\mathbf{R}_B^{(k)}(t) \\
=&\begin{bmatrix} \beta^{1/2}\mathbf{x}(t) & \alpha^{1/2}\hat{\mathbf{X}}_{L-1}^{(K)H}(t-1) \end{bmatrix} \hat{\mathbf{B}}^{(k)}(t) \\
=&\begin{bmatrix} \beta^{1/2}\mathbf{x}(t) & \alpha^{1/2}\hat{\mathbf{X}}^{(K)H}(t-1) \end{bmatrix} \begin{bmatrix} \beta^{1/2}\mathbf{g}^{(k-1)H}(t) \\ \alpha^{1/2}\mathbf{Q}_B^{(0)}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t) \end{bmatrix} \\
&-\left(\alpha^{1/2}\hat{\mathbf{X}}^{(K)H}(t-1)\mathbf{z}_L\right)\left(\alpha^{1/2}\mathbf{z}_L^T\mathbf{Q}_B^{(0)}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t)\right) \qquad (4.49) \\
=&\alpha\mathbf{Q}_A^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t)+\beta\mathbf{x}(t)\mathbf{g}^{(k-1)H}(t) \\
&-\alpha\mathbf{Q}_A^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{q}_{B_L}(t)\mathbf{q}_{B_L}^H(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t) \qquad (4.50) \\
=&\alpha\mathbf{Q}_A^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t)+\beta\mathbf{x}(t)\mathbf{g}^{(k-1)H}(t) \\
&-\alpha\mathbf{Q}_A^{(0)}(t)\mathbf{p}(t)\mathbf{p}^H(t)\mathbf{\Theta}_A^{(k-1)}(t) \qquad (4.51)
\end{aligned}
$$

where we replace $\hat{\mathbf{B}}^{(k)}(t)$ with (4.12) where $\mathbf{R}_A^{(k)H}(t)$ is replaced by $\mathbf{R}_B^{(k)}(t)$ in (4.49) and $\hat{\mathbf{X}}^{(K)}(t-1)$ with (4.46) in (4.50) and let $\mathbf{p}(t)=\mathbf{R}_B^{(0)H}(t)\mathbf{q}_{B_L}(t)$. Note that $\hat{\mathbf{X}}^{(K)H}(t-1)\mathbf{z}_L = \mathbf{Q}_A^{(0)}(t)\mathbf{p}(t)$ is the conjugate transpose of the last row of $\hat{\mathbf{X}}^{(K)}(t-1)$.

Similar to the operations on $\mathbf{z}$ in the first QR decomposition, we decompose the data vector $\mathbf{x}(t)$ into a component that lies in the "old" principal subspace spanned by the

columns of $\bar{\mathbf{Q}}_A^{(0)}(t)$ and an orthogonal innovation vector $\mathbf{x}_\perp(t)$ and can obtain

$$\mathbf{x}_\perp(t) = \left(\mathbf{I} - \bar{\mathbf{Q}}_A^{(0)}(t)\bar{\mathbf{Q}}_A^{(0)H}(t)\right)\mathbf{x}(t) = \mathbf{x}(t) - \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t). \tag{4.52}$$

where $\bar{\mathbf{Q}}_A^{(0)}(t)\bar{\mathbf{Q}}_A^{(0)}(t)$ is the orthogonal projector onto $\mathcal{R}(\bar{\mathbf{Q}}_A^{(0)}(t))$. By normalizing $\mathbf{x}_\perp(t)$, we can obtain

$$\mathbf{x}(t) = \mu_x(t)\bar{\mathbf{x}}_\perp(t) + \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t) \tag{4.53}$$

where $\mu_x(t) = (\mathbf{x}_\perp^H(t)\mathbf{x}_\perp(t))^{1/2}$ and $\bar{\mathbf{x}}_\perp(t) = \mathbf{x}_\perp(t)/\mu_x(t)$. Since all the components of $\mathbf{x}(t)$, $\mu_x(t)$, $\bar{\mathbf{x}}_\perp(t)$, $\mathbf{h}^{(0)}(t)$, $\bar{\mathbf{Q}}_A^{(0)}(t)$, are not changing with different inner iteration index k, they also only need to be computed once at each time instant. Moreover, due to the fact that $\mathbf{A}^{(k)}(t) = \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{R}_A^{(k)}(t)$ as in Step 8 of Algorithm 4.3, replacing $\mathbf{x}(t)$ by (4.53) in (4.51) we have

$$\begin{aligned}
&\bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{R}_A^{(k)}(t)\mathbf{R}_B^{(k)}(t) \\
=&\alpha\bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{U}^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \beta\left(\mu_x(t)\bar{\mathbf{x}}_\perp(t) + \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t)\right)\mathbf{g}^{(k-1)H}(t) \\
&- \alpha\bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{U}^{(0)}(t)\mathbf{p}(t)\mathbf{p}^H(t)\mathbf{\Theta}_A^{(k-1)}(t) \\
=&\begin{bmatrix}\bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t)\end{bmatrix}\begin{bmatrix}\alpha\mathbf{U}^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \beta\mathbf{h}^{(0)}(t)\mathbf{g}^{(k-1)H}(t) - \alpha\mathbf{s}(t)\mathbf{r}^{(k-1)H}(t) \\ \beta\mu_x(t)\mathbf{g}^{(k-1)H}(t)\end{bmatrix}
\end{aligned} \tag{4.54}$$

where $\mathbf{s}(t) = \mathbf{U}^{(0)}(t)\mathbf{p}(t)$ is constant at each time instant and $\mathbf{r}^{(k-1)}(t) = \mathbf{\Theta}_A^{(k-1)H}(t)\mathbf{p}(t)$. Similar to $\mathbf{G}_B^{(k)}(t)$, let $\mathbf{G}_A^{(k)}(t)$ be a sequence of $(r'^2/2 + r'/2)$ orthonormal Givens rotations

that converts the second matrix on the right handside of (4.54) to an upper triangular matrix $\mathbf{R}^{(k)}(t)$ with an additional zero row. Then, we can obtain that

$$\left[\bar{\mathbf{Q}}_A^{(k)}(t) \quad \mathbf{q}_{A\perp}^{(k)}(t)\right] = \left[\bar{\mathbf{Q}}_A^{(0)}(t) \quad \bar{\mathbf{x}}_\perp(t)\right] \mathbf{G}_A^{(k)H}(t) \tag{4.55}$$

$$\begin{bmatrix} \mathbf{R}^{(k)}(t) \\ 0 \dots 0 \end{bmatrix} = \mathbf{G}_A^{(k)}(t) \begin{bmatrix} \alpha \mathbf{U}^{(0)}(t) \mathbf{R}_B^{(0)H}(t) \mathbf{R}_B^{(0)}(t) \mathbf{\Theta}_A^{(k-1)}(t) + \beta \mathbf{h}^{(0)}(t) \mathbf{g}^{(k-1)H}(t) - \alpha \mathbf{s}(t) \mathbf{r}^{(k-1)H}(t) \\ \beta \mu_x(t) \mathbf{g}^{(k-1)H}(t) \end{bmatrix}. \tag{4.56}$$

Then, similar to (4.47) the updated $\bar{\mathbf{Q}}_A^{(k)}(t)$ could be obtained from (4.55) by rotating $[\tilde{\mathbf{Q}}_A^{(0)}(t), \bar{\mathbf{x}}_\perp(t)]$ using the sequence of Givens matrix $\mathbf{G}_A^{(k)H}(t)$ which requires about $2Nr'(r'+1)$ flops. Since $\mathbf{R}^{(k)}(t) = \mathbf{R}_A^{(k)}(t)\mathbf{R}_B^{(k)}(t)$, $\mathbf{R}_A^{(k)}(t)$ could be computed via r' by r' back substitution. Furthermore, we can have

$$\begin{aligned} \mathbf{G}_A^{(k)H}(t) &= \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)H}(t) \\ \bar{\mathbf{x}}_\perp^H(t) \end{bmatrix} \left[\bar{\mathbf{Q}}_A^{(k)}(t) \quad \mathbf{q}_{A\perp}^{(k)}(t)\right] \\ &= \begin{bmatrix} \bar{\mathbf{\Theta}}_A^{(k)}(t) & \bar{\mathbf{Q}}_A^{(0)H}(t) \mathbf{q}_\perp^{(k)}(t) \\ \bar{\mathbf{x}}_\perp^H(t) \bar{\mathbf{Q}}_A^{(k)}(t) & \bar{\mathbf{x}}_\perp^H(t) \mathbf{q}_{A\perp}^{(k)}(t) \end{bmatrix}. \end{aligned} \tag{4.57}$$

Then, $\bar{\mathbf{\Theta}}_A^{(k)}(t)$ can be directly extracted from $\mathbf{G}_A^{(k)H}(t)$ as a by-product as mentioned earlier and used in the computation of the first QR decomposition at the $(k+1)$th inner iteration with little effort.

If we let $\bar{\mathbf{Q}}_A^{(k)}(t)$ replaced by $\mathbf{Q}_A^{(k)}(t)$ and $\mathbf{U}^{(k)}(t)$ be constantly an identity matrix during each time instant, so far the extension of bi-iteration SVD algorithm to multiple

inner iterations but without Ritz acceleration is already described for Algorithm 4.3, called Bi-SVD2-M-1.

4.4.3 Ritz Acceleration

Similar to the first QR decomposition, the derivations of integrating Ritz acceleration into bi-iteration SVD algorithm for this proposed algorithm is again almost the same as that for Bi-SVD2-MR-1. So we only show the results as follows

$$\begin{aligned}\hat{\mathbf{T}}^{(k)}(t) =& \alpha \left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{R}_B^{(0)H}(t)\right)\left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{R}_B^{(0)H}(t)\right)^H \\ & - \alpha \left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{p}(t)\right)\left(\tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{p}(t)\right)^H + \beta \mathbf{h}^{(k)}(t)\mathbf{h}^{(k)H}(t) \end{aligned} \tag{4.58}$$

where $\mathbf{p}(t)$ is defined after (4.51),

$$\mathbf{h}^{(k)}(t) = \bar{\mathbf{Q}}_A^{(k)H}(t)\mathbf{x}(t) \tag{4.59}$$

will be used in the first QR decomposition at the $(k+1)$th inner iteration and

$$\tilde{\mathbf{\Theta}}_A^{(k)}(t) = \mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k)}(t) \tag{4.60}$$

be used to compute $\bar{\mathbf{\Theta}}_A^{(k)}(t)$ later so that $\mathbf{\Theta}_A^{(k)}(t) = \tilde{\mathbf{\Theta}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$ in the counterpart of (4.13). Then, the ED of $\hat{\mathbf{T}}^{(k)}(t)$ can be computed by

$$\hat{\mathbf{T}}^{(k)}(t) = \mathbf{U}^{(k)}(t)\mathbf{D}^{(k)}(t)\mathbf{U}^{(k)H}(t). \tag{4.61}$$

which requires $O(r'^3)$ flops and can be obtained very efficiently by symmetric QR algorithm. Therefore, $\mathbf{Q}^{(k)}(t)$ can be expressed by the first r columns of $\mathbf{Q}_A^{(k)}(t) \triangleq \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$. Again, when we have a large enough K, $\mathbf{U}^{(K)}(t)$ will be very close to an identity matrix so that the algorithm complexity could be further reduced.

4.4.4 Algorithm Description, Bi-SVD2-MR-1

The overall linear complexity version of Algorithm 4.3 with Ritz acceleration and multiple inner iterations with complexity $O(Nr'^2)+O(Lr'^2)$ for subspace tracking could be described as in Algorithm 4.5 as Bi-SVD2-MR-1.

Algorithm 4.5 (Bi-SVD2-MR-1)**.**

1: ***Initialization:*** $\bar{\mathbf{Q}}_A^{(K)}(0) = [\mathbf{I}_{r'}, \mathbf{0}]^T$, $\mathbf{U}^{(K)}(0) = \mathbf{I}_{r'}$, $\mathbf{U}^{(K)}(0) = \mathbf{I}_{r'}$, $\mathbf{R}_A^{(K)}(0) = \mathbf{I}_{r'}$, *and* $\mathbf{Q}_B^{(K)}(0) = [\mathbf{I}_{r'}, \mathbf{0}]^T$, $\mathbf{U}^{(K)}(0) = \mathbf{I}_{r'}$. *Let* $\mathbf{z} = [1, 0, \ldots, 0]^T$, $\mathbf{z}_L = [0, \ldots, 0, 1]^T$.

2: ***for*** $t = 1, 2, \ldots$ ***do***

3: ***Initialization:*** *: Let* $\bar{\mathbf{Q}}_A^{(0)}(t) = \bar{\mathbf{Q}}_A^{(K)}(t-1)$, $\mathbf{U}^{(0)}(t) = \mathbf{U}^{(K)}(t-1)$, $\bar{\mathbf{\Theta}}_A^{(0)}(t) = \mathbf{I}_{r'}$, $\mathbf{R}_B^{(0)}(t) = \mathbf{R}_B^{(K)}(t-1)$, *and* $\mathbf{Q}_B^{(0)}(t) = \mathbf{Q}_B^{(K)}(t-1)$.

4: $\mathbf{h}^{(0)}(t) = \bar{\mathbf{Q}}_A^{(0)H}(t)\mathbf{x}(t)$, $\mathbf{q}_{B_L}(t) = \mathbf{Q}_B^{(0)H}(t)\mathbf{z}_L$, $\mathbf{p}(t) = \mathbf{R}_B^{(0)H}(t)\mathbf{q}_{B_L}(t)$, $\mathbf{s}(t) = \mathbf{U}^{(0)}(t)\mathbf{p}(t)$ —$Nr' + 2r'^2$

5: $\tilde{\mathbf{Q}}_B^{(0)}(t) = \begin{bmatrix} \mathbf{0}^H & 1 \\ \mathbf{I}_{L-1} & \mathbf{0} \end{bmatrix} \mathbf{Q}_B^{(0)}(t)$

6: $\mathbf{z}_\perp(t) = \mathbf{z} - \tilde{\mathbf{Q}}_B^{(0)}(t)\mathbf{q}_{B_L}(t)$, $\mu_z(t) = (\mathbf{z}_\perp^H(t)\mathbf{z}_\perp(t))^{1/2}$, $\bar{\mathbf{z}}_\perp(t) = \mathbf{z}_\perp(t)/\mu_z(t)$ —Lr'

7: $\mathbf{x}_\perp(t) = \mathbf{x}(t) - \bar{\mathbf{Q}}_A^{(0)}(t)\mathbf{h}^{(0)}(t)$, $\mu_x(t) = (\mathbf{x}_\perp^H(t)\mathbf{x}_\perp(t))^{1/2}$, $\bar{\mathbf{x}}_\perp(t) = \mathbf{x}_\perp(t)/\mu_x(t)$ —Nr'

8: ***for*** $k = 1, 2, \ldots, K$ ***do***

9: ***First iteration:***

10: $\mathbf{g}^{(k-1)}(t) = \mathbf{U}^{(k-1)H}(t)\mathbf{h}^{(k-1)}(t)$ —r'^2

11: $\tilde{\mathbf{g}}^{(k-1)}(t) = \beta^{1/2}\mathbf{g}^{(k-1)}(t) - \alpha^{1/2}\mathbf{\Theta}_A^{(k-1)H}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{q}_{B_L}(t)$ —$2r'^2$

12: $\begin{bmatrix} \mathbf{R}_B^{(k)}(t) \\ \\ 0 \dots 0 \end{bmatrix} = \mathbf{G}_B^{(k)}(t) \begin{bmatrix} \alpha^{1/2}\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t) + \mathbf{q}_{B_L}(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \\ \\ \mu_z(t)\tilde{\mathbf{g}}^{(k-1)H}(t) \end{bmatrix}$ — $r'^3 + r'^2 + 2r'(r'+1)^2$

13: ***if*** $k = K$ ***then***

14: $\begin{bmatrix} \mathbf{Q}_B^{(k)}(t) & \mathbf{q}_{B\perp}^{(k)}(t) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{Q}}_B^{(0)}(t) & \bar{\mathbf{z}}_\perp(t) \end{bmatrix} \mathbf{G}_B^{(k)H}(t)$ —*about* $2Lr'^2$ *only if* $k = K$

15: ***end if***

16: ***Second iteration:***

17: $\mathbf{r}^{(k-1)}(t) = \mathbf{\Theta}_A^{(k-1)H}(t)\mathbf{p}(t)$ —r'^2

18: $\mathbf{P}^{(k-1)}(t) = \mathbf{U}^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t)$ —$3r'^3$

19: $\begin{bmatrix} \mathbf{R}^{(k)}(t) \\ \\ 0 \dots 0 \end{bmatrix} = \mathbf{G}_A^{(k)}(t) \begin{bmatrix} \alpha\mathbf{P}^{(k-1)}(t) + \beta\mathbf{h}^{(0)}(t)\mathbf{g}^{(k-1)H}(t) - \alpha\mathbf{s}(t)\mathbf{r}^{(k-1)H}(t) \\ \\ \beta\mu_x(t)\mathbf{g}^{(k-1)H}(t) \end{bmatrix}$ — $2r'^2 + 2r'(r'+1)^2$

20: *Compute* $\mathbf{R}_A^{(k)}(t)$ *from* $\mathbf{R}^{(k)}(t) = \mathbf{R}_A^{(k)}(t)\mathbf{R}_B^{(k)}(t)$ *via backsubstitution.*

21: ***if*** $k = K$ ***then***

22: $\begin{bmatrix} \bar{\mathbf{Q}}_A^{(k)}(t) & \mathbf{q}_{A\perp}^{(k)}(t) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t) \end{bmatrix} \mathbf{G}_A^{(k)H}(t)$ —*about* $2Nr'^2$ *only if* $k = K$

23: ***end if***

24: *Extract* $\bar{\mathbf{\Theta}}_A^{(k)}(t)$ *from* $\mathbf{G}_A^{(k)}(t)$ *by* (4.57)

25: ***Ritz Acceleration:***

26: $\mathbf{h}^{(k)}(t) = \bar{\mathbf{G}}_A^{(k)}(t) \begin{bmatrix} \mathbf{h}^{(0)}(t) \\ \\ \bar{\mathbf{x}}_\perp^H(t)\mathbf{x}(t) \end{bmatrix}$ —$r'^2 + r'$

27: $\tilde{\mathbf{\Theta}}_A^{(k)}(t) = \mathbf{U}^{(0)H}(t)\bar{\mathbf{\Theta}}_A^{(k)}(t)$ —r'^3

28: $\mathbf{S}^{(k)}(t) = \tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{R}_B^{(0)H}(t)$, $\mathbf{w}^{(k)}(t) = \tilde{\mathbf{\Theta}}_A^{(k)H}(t)\mathbf{p}(t)$ —$r'^3 + r'^2$

29: $\hat{\mathbf{T}}^{(k)}(t) = \alpha\mathbf{S}^{(k)}(t)\mathbf{S}^{(k)H}(t) - \alpha\mathbf{w}(t)\mathbf{w}^H(t) + \beta\mathbf{h}^{(k)}(t)\mathbf{h}^{(k)H}(t)$ —$r'^3 + 2r'^2$

30: $\hat{\mathbf{T}}^{(k)}(t) = \mathbf{U}^{(k)}(t)\mathbf{D}^{(k)}(t)\mathbf{U}^{(k)H}(t)$, *ED* —$O(r'^3)$

31: ***end for***

32: *Let* $\mathbf{Q}^{(k)}(t)$ *be the first* r *columns of* $\mathbf{Q}_A^{(k)}(t) \triangleq \bar{\mathbf{Q}}_A^{(k)}(t)\mathbf{U}^{(k)}(t)$ *or* $\bar{\mathbf{Q}}_A^{(k)}(t)$ —Nr'^2 *or none*

33: ***end for***

In the above algorithm, similar modifications are made as in Section 4.3.4. Actually, $\mathbf{Q}_B^{(k)}(t)$, $k = 1, \ldots, K$, is never used except when transferred to the next time instant $t+1$ for initialization as $\mathbf{Q}_B^{(K)}(t)$. Thus, only about $2Lr'(r'+1)$ flops to compute $\mathbf{Q}_B^{(K)}(t)$ in (4.47) are required at the Kth inner iteration as in Step 14 no matter how many inner iterations are assigned for each data update. $\mathbf{Q}_A^{(k)}(t)$, $k = 1, \ldots, K$, only appears in (4.59) in the second QR decomposition during each data update other than being transferred to $(t+1)$ for initialization as $\mathbf{Q}_A^{(K)}(t)$ and could be rewritten as

$$\begin{aligned}\mathbf{h}^{(k)}(t) =& \bar{\mathbf{Q}}_A^{(k)H}(t)\mathbf{x}(t) \\ =& \bar{\mathbf{G}}_A^{(k)}(t)\begin{bmatrix}\bar{\mathbf{Q}}_A^{(0)}(t) & \bar{\mathbf{x}}_\perp(t)\end{bmatrix}^H \mathbf{x}(t) \\ =& \bar{\mathbf{G}}_A^{(k)}(t)\begin{bmatrix}\mathbf{h}^{(0)}(t) \\ \bar{\mathbf{x}}_\perp^H(t)\mathbf{x}(t)\end{bmatrix}\end{aligned} \tag{4.62}$$

where $\bar{\mathbf{G}}_A^{(k)}(t) = [\mathbf{I}_{r'}, \mathbf{0}]\mathbf{G}_A^{(k)}(t)$ consists of the first r' rows of $\mathbf{G}_A^{(k)}(t)$ and the corresponding

computations are reduced from Nr' to $r'^2 + r'$. Furthermore, only $2Nr'(r'+1)$ flops are required to compute $\mathbf{Q}_A^{(K)}(t)$ as the initialization transferred to $t+1$ in (4.55) once at the Kth inner iteration no matter how large K is. Then, all the steps in Bi-SVD2-MR-1 have linear complexity of $O(Nr') + O(Lr')$ or less $O(r'^3)$ when $N \gg r'$ so that $N > r'^2$, except Step 14 and 22 of about $2Nr'^2 + 2Lr'^2$ but only happen once for each data update as in Bi-SVD1-MR-1. Again, $\mathbf{U}^{(0)}(t)$ could be replaced by $\mathbf{I}_r$ and removed from the proposed algorithms so that another $2r'^3$ could be further reduced from Step 18 and 27 as it will converge to an identity matrix when K is large enough.

4.4.5 Ultrafast Version, Bi-SVD2-MR-2

As in Section 4.3.5, when N is large, r' could also be a large dimension and the steps with $O(Nr'^2)$ or $O(Lr'^2)$ could consume more computations than all the other algorithm steps. For faster tracking algorithms, we could take the upper Hessenberg part with a number of $l < r'$ subdiagonals below the main diagonal of one component in (4.48) and (4.56) so that only a sequence of about $(l+2)r'$ Givens rotations for (4.48) or about $(l+3)r'$ Givens rotations for (4.56) is required to explicitly compute $\mathbf{Q}_B^{(K)}(t)$ and $\mathbf{Q}_A^{(K)}(t)$. Specifically, for (4.48) all the steps are the same except that $\mathbf{R}_A^{(0)H}(t)$ needs to be replaced $\mathbf{R}_B^{(0)}(t)$ and are omitted here. Unlike in Bi-SVD1-MR-2, to compute $\mathbf{Q}_A^{(K)}(t)$ in (4.56), we take the upper Hessenburg part of $\mathbf{U}^{(0)}(t)\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)\mathbf{\Theta}_A^{(k-1)}(t)$ and the second matrix on the right hand side of (4.56) could be rewritten as the sum of an upper Hessenberg plus one additional row with all zeros and two rank one matrices. Similar procedure could be operated to convert it to an upper triangular matrix. Again, in this way the main diagonal

and upper rows with most weights are preserved. Instead of the lower triangular matrix $\mathbf{R}_B^{(1)H}(t)$, the near diagonal matrix $\mathbf{R}_B^{(0)H}(t)\mathbf{R}_B^{(0)}(t)$ is involved and suppresses the lower rows of $\mathbf{\Theta}_A^{(k-1)}(t)$ so that extracting the upper Hessenburg part will not lead to much distortion.

To keep the near diagonal property of $\mathbf{R}_B^{(0)}(t)$ for better approximation by taking the upper Hessenburg part of certain matrix component, we let $\mathbf{R}_B^{(0)}(t) = \mathbf{R}_B^{(K-1)}(t-1)$ as the initializations at each time instant. Thus, Step 14 and 22 of Bi-SVD2-MR-1 will be further reduced and the corresponding lower complexity algorithm is called, Bi-SVD2-MR-2. The same simplifications could also be applied to Bi-SVD2-M-1 and produces its faster version, Bi-SVD2-M-2.

4.5 Simulation Results

In this section, we show the simulation results of all the proposed subspace tracking algorithms, Bi-SVD1-M-1 and Bi-SVD1-MR-1, Bi-SVD2-M-1 and Bi-SVD2-MR-1, and their ultrafast version, Bi-SVD1-M-2 and Bi-SVD1-MR-2, Bi-SVD2-M-2 and Bi-SVD2-MR-2. Bi-LS algorithm in [44] with single inner iteration and no Ritz acceleration is used as the reference for comparison. Note that although having almost the same algorithm performance as [58] on which our proposed algorithm is based and with even lower complexity, Bi-LS algorithm cannot be equipped with multiple inner iterations and Ritz acceleration for an enlarged tracking dimension. This is because unlike in [58] the estimated matrices in Bi-LS essentially converge to the principal singular vectors times a unitary matrix and the order between the columns of the estimated matrices converging to the principal singular vectors in [58] is lost due to multiplication by the unitary matrix.

4.5.1 Setups

The test data are chosen to be the same as in [58] [44]:

$$x(t) = \sum_{m=1}^{M} \sin(2\pi f_m t) + w(t) \tag{4.63}$$

which is a sum of M real sinusiod sequences plus white noise $w(t)$ with variance σ^2. Thus, the subspace or rank parameter r is fixed at twice of M, i.e. $r = 2M$. $SNR = \frac{M}{2\sigma^2}$. Updated data vector is given by $\mathbf{x}(t) = [x(t), x(t+1), \ldots, x(t+N-1)]^H$. We consider the "jump scenario" where frequencies of each sinusoid are constant in a segment of duration $\mathbf{T}_p$ samples, $p = 1, \ldots, P$, and P defines the total number of segments. They change abruptly at the segment boundaries.

In the following simulation, data are generated when $N = 80$, $M = 2$ ($r = 4$), $L = 100$, $\alpha = 0.95$, $\beta = 0.05$, $SNR = 30$dB. There are $P = 2$ segments each having $T_1 = T_2 = 300$ samples. Thus, $2r = 8$-dimensional principal subspace exists during the transition period starting from $t = 301$ to $t = 400$. The frequencies for the first segment are $f_1 = 0.0517$ and $f_2 = 0.3583$ and changed to $f_1 = 0.3455$ and $f_2 = 0.1646$ during the second segment. The algorithm performance is measured by the subspace distance defined in (4.1) between the one spanned by the columns of estimated $\mathbf{Q}^{(k)}(t)$ and the true subspace of $\mathbf{X}(t)$ given by SVD.

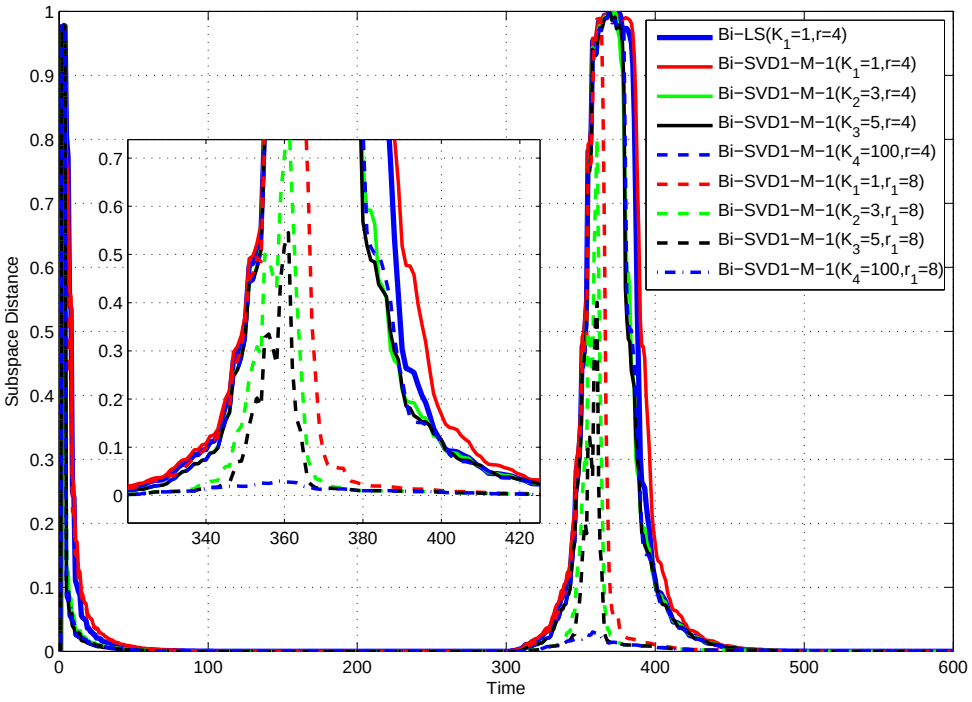

Figure 4.1: Bi-SVD1-M-1 algorithm. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

4.5.2 Multiple Inner Iterations, Bi-SVD1-M-1 and Bi-SVD2-M-1, for r and Enlarged r'

First of all, we apply Bi-SVD1-M-1 and Bi-SVD2-M-1 to evaluate the impact of multiple inner iterations but without Ritz acceleration as in Fig. 4.1 and 4.2. We can see that in Fig. 4.1 for Bi-SVD1-M-1, when $r = 4$, as the number of inner iterations K increases from $K_1 = 1$ to $K_4 = 100$, the distance between estimated subspace and true subspace could only be reduced slightly and cannot even be differentiated from the corresponding curves during some periods. Even a very large $K_4 = 100$ produces almost the same subspace distance curves as $K_2 = 3$ and fails to provide further improvement although many more

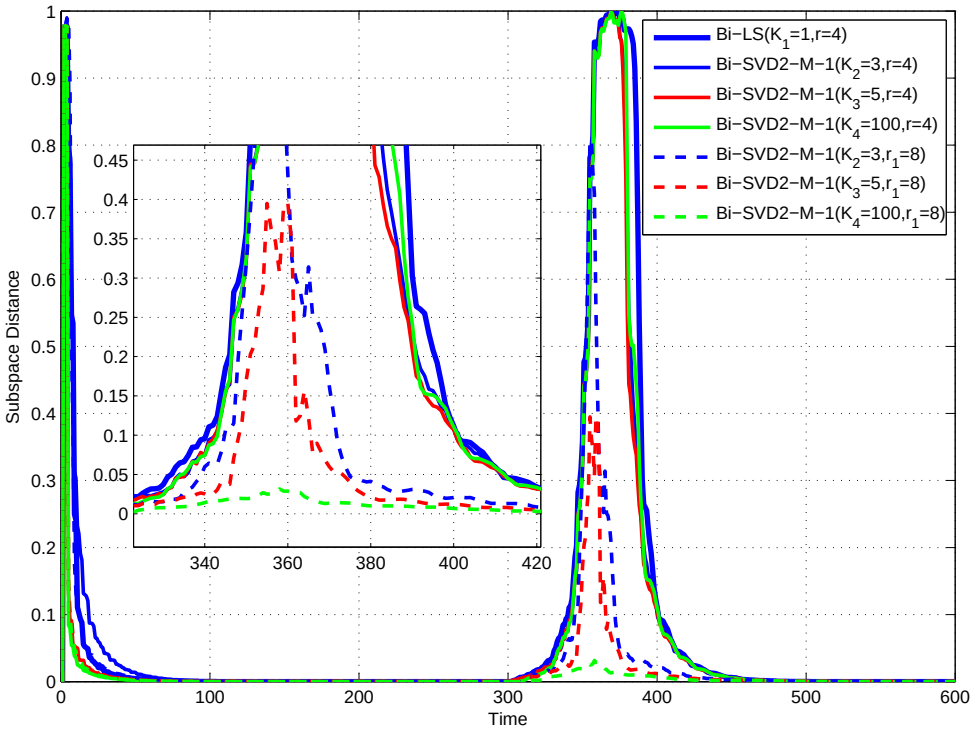

Figure 4.2: Bi-SVD2-M-1 algorithm. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

inner iterations are consumed. This indicates that the subspace tracking accuracy could not be greatly enhanced by increasing the number of multiple inner iterations while keeping $r = 4$. The simulation results confirm our analysis in Section 4.2 that tracking only r-dimensional subspace generates a poor data matrix approximation in (4.10) with its own principal subspace not quite close to the true subspace of real data matrix $\mathbf{X}(t)$. Thus, although many inner iterations are committed, they only drive the estimated principal subspace towards the wrong subspace from the inaccurate data approximation, still having large distance from the true principal subspace. For Bi-SVD2-M-1, as we mentioned that

$K_1 = 1$ is not eligible. So we only show $K_2 = 3$ to $K_4 = 100$ in Fig. 4.2 where the same conclusions could be made as for Bi-SVD1-M-1.

On the other hand, when tracking an enlarged dimension of the $r' \geq 2r$ principal right singular vectors but only the first r of them are of interest, all the principal right singular vectors could be included in a much more accurate data matrix approximation of (4.45) and better tracking performance is highly expected. This is exactly what is shown with $r_1 = 2r = 8$ in Fig. 4.1 and 4.2. Given this enlarged tracking dimension, both Bi-SVD1-M-1 and Bi-SVD2-M-1 algorithms display much improved tracking performance compared to $r = 4$. As K increases, more accurate principal subspace could be obtained at each time instant and with a quite large $K_4 = 100$, the distance between the estimated subspace and the true principal subspace becomes extremely small even during the whole transition period. When taking a more careful look on $K_3 = 5$, we can see that the maximum subspace distance for Bi-SVD1-M-1 is only about 0.55 and for Bi-SVD2-M-1 is less than 0.4 during a few time instants while that of Bi-LS is 1 and lasts for over 20 time instants. Also, it only has about 25 time instants after $t = 300$ during which period Bi-SVD1-M-1 yields subspace distances larger than 0.02 while Bi-LS takes about 150 time instants for the same criterion. For Bi-SVD2-M-1, the time instants for the same period is about 40. Thus, due to the much more accurate data matrix approximation by tracking enlarged dimension subspace, Bi-SVD1-M-1 and Bi-SVD2-M-1 algorithm shows greatly improved performance in terms of both tracking accuracy and convergence rate when K equals a relatively small number of 3-5. While Bi-SVD1-M-1 has a shorter period when the subspace distance is higher than a threshold of 0.02, Bi-SVD2-M-1 produces smaller maximum value during the

whole transition period.

4.5.3 Ritz Acceleration, Bi-SVD1-MR-1 and Bi-SVD2-MR-1

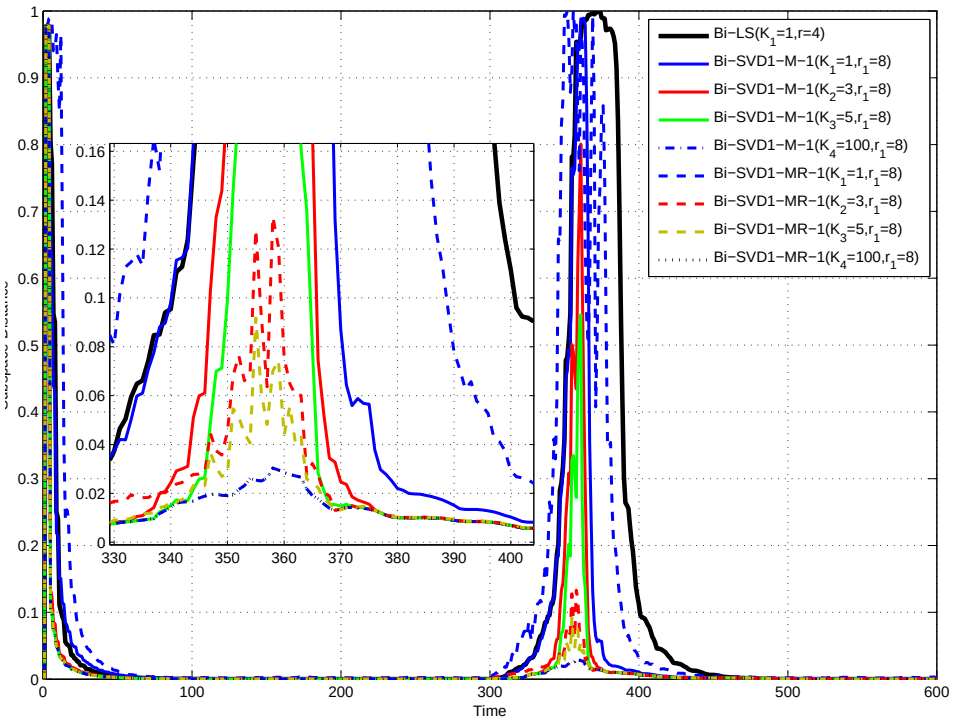

Figure 4.3: Bi-SVD1-MR-1 algorithm. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

In this part the performance of bi-iteration SVD algorithm with both multiple inner iterations and Ritz acceleration, Bi-SVD1-MR-1 and Bi-SVD2-MR-1, for enlarged tracking dimension $r1 = 8$ are evaluated as in Fig. 4.3 and 4.4. It can be shown that when equipped with Ritz acceleration, both algorithms further improve the performance

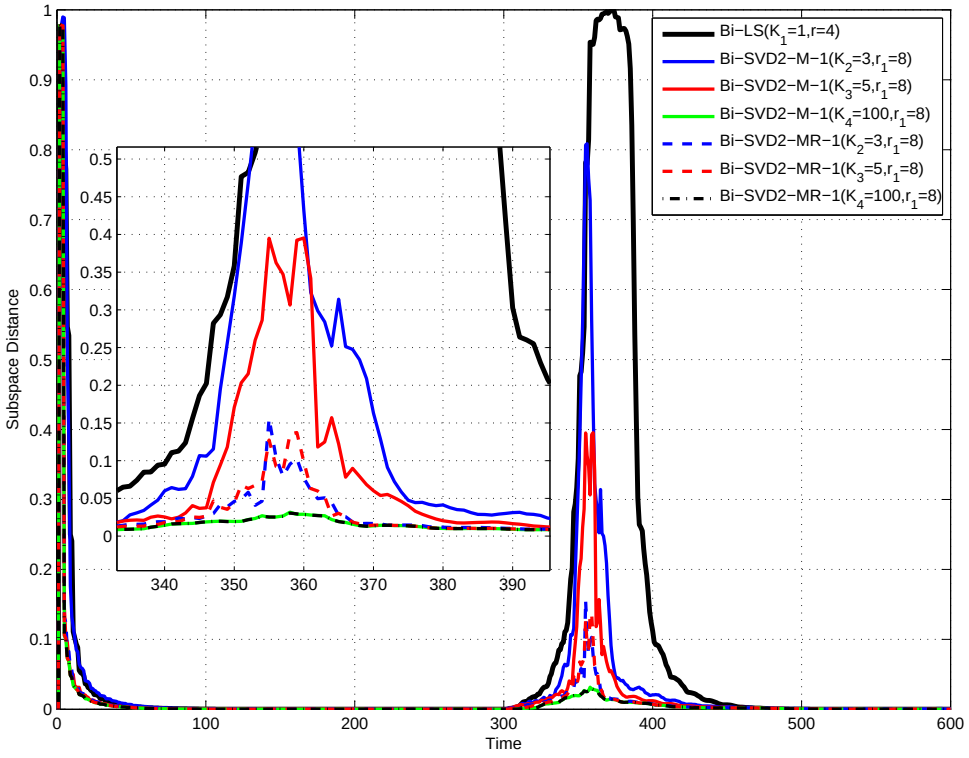

Figure 4.4: Bi-SVD2-MR-1 algorithm. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

for both tracking accuracy and convergence rate. The improvement is so impressive that the maximum subspace distance for even a small number of inner iterations $K_2 = 3$ is close to 0.13 for Bi-SVD1-MR-1 and 0.15 for Bi-SVD2-MR-1, not to mention that for $K_3 = 5$ and $K_4 = 100$ the maximum values are even smaller. The number of time instants with a subspace distance larger than 0.02 during the transition period is only about 25 for all $K > 1$, not only much smaller than that of Bi-LS but also than that of Bi-SVD1-M-1 and Bi-SVD2-M-1. The exceptional performance of algorithms equipped with Ritz acceleration is not surprising considering that the convergence rate of the estimated r principal right

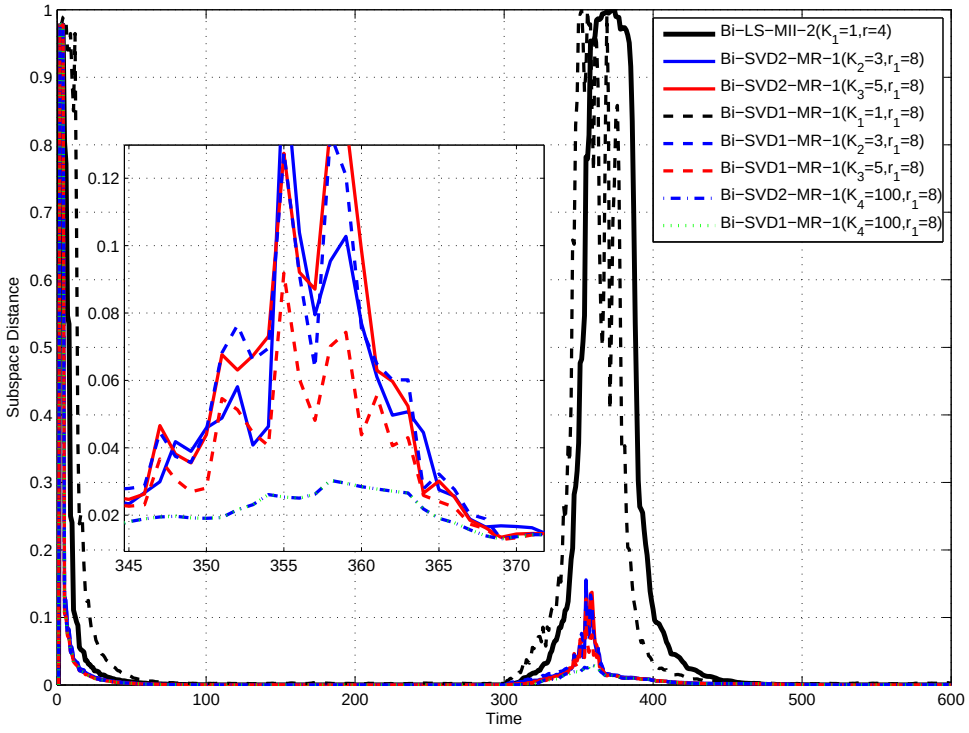

Figure 4.5: Bi-SVD1-MR-1 and Bi-SVD2-MR-1 algorithms. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

singular vectors is $(|\lambda_{r_1+1}|/|\lambda_r|)^{2k}$ while that of algorithms without Ritz acceleration is $(|\lambda_{r+1}|/|\lambda_r|)^{2k}$ as analyzed in Section 4.2.1. Since as a minor singular value $|\lambda_{r_1+1}|$ is much smaller than $|\lambda_{r+1}|$, the convergence rate of Bi-SVD1-MR-1 and Bi-SVD2-MR-1 is much larger than that of Bi-SVD1-M-1 and Bi-SVD2-M-1. Meanwhile, when $K_4 = 100$, algorithms with and without Ritz acceleration have almost the same subspace distance curves as they overlap, displaying that with extremely large K algorithms withtout Ritz acceleration could also produce the same performance as those with Ritz but with much more computations. Therefore, the benefits of Ritz acceleration is highlighted as Bi-SVD1-

MR-1 and Bi-SVD2-MR-1 only needs a few inner iterations (about 3-5) to accomplish the same performance for Bi-SVD1-M-1 and Bi-SVD2-M-1 with many more inner iterations and saves a large amount of computations. The comparison of Bi-SVD1-MR-1 and Bi-SVD2-MR-1 is shown in Fig. 4.5

4.5.4 Ultrafast Subspace Tracking Algorithm, Bi-SVD2-M-2 and Bi-SVD2-MR-2

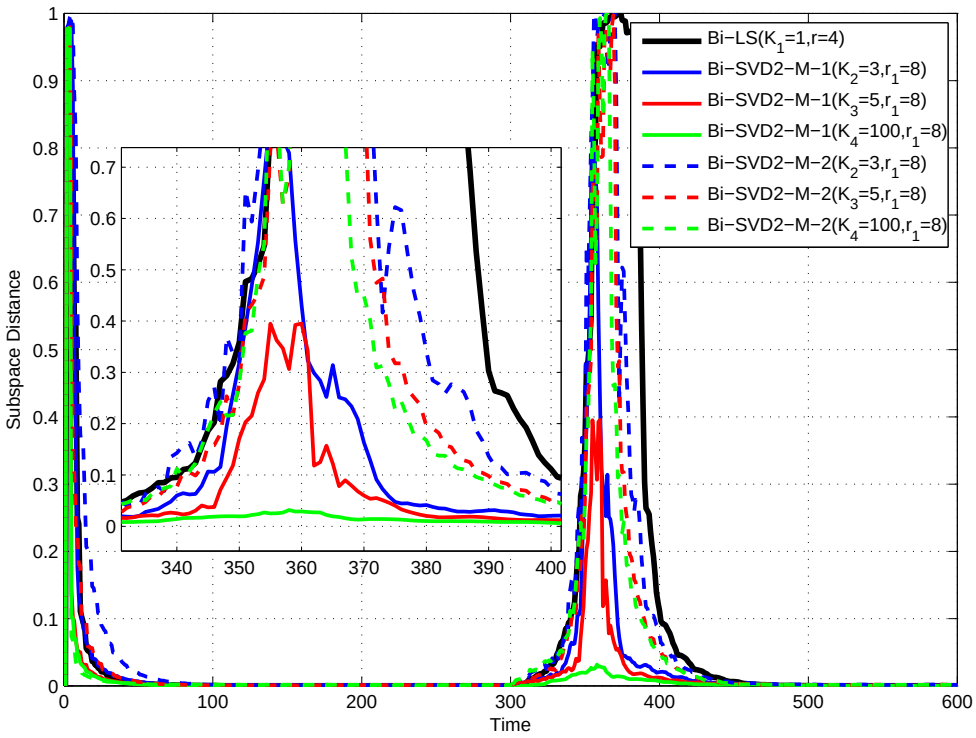

Figure 4.6: Bi-SVD2-M-1 and Bi-SVD2-M-2 algorithms. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

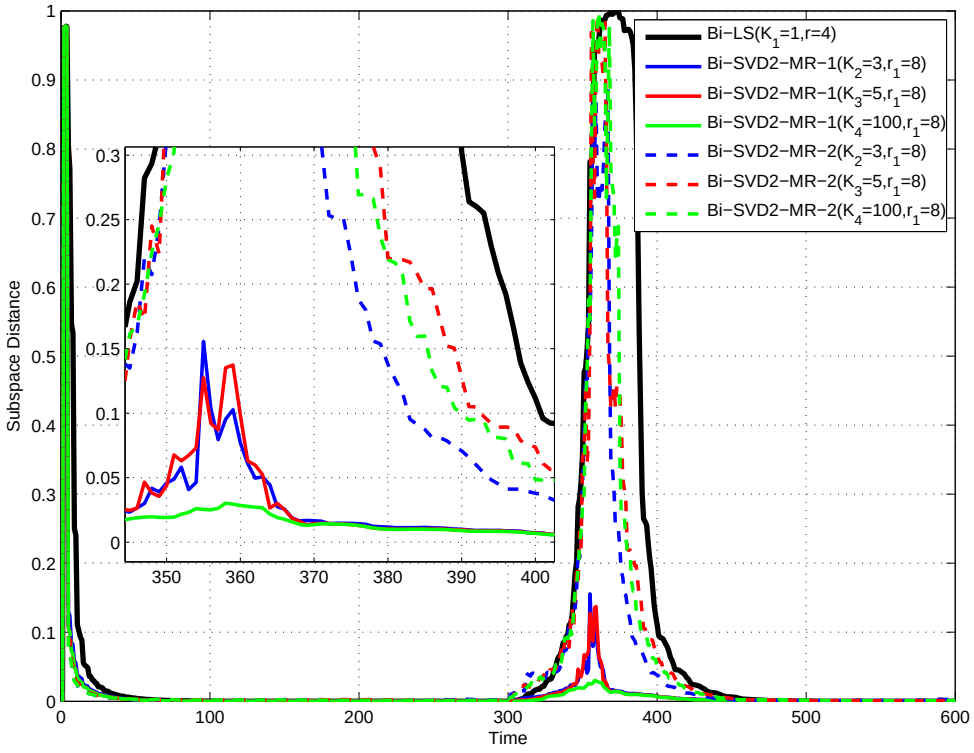

Figure 4.7: Bi-SVD2-MR-1 and Bi-SVD2-MR-2 algorithms. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

If r' is large, we may prefer the ultrafast subspace tracking algorithms, Bi-SVD2-M-2 and Bi-SVD2-MR-2, when computational limits are imposed. Their simulation results compared to Bi-SVD2-M-1 and Bi-SVD2-MR-1 are displayed in Fig. 4.6 and 4.7. Although not as good as the performance of Bi-SVD2-M-1 and Bi-SVD2-MR-1, they still outperform Bi-LS algorithm with the same level of complexity of $O(Nr') + O(Lr')$. We can see that as K becomes quite large, the algorithm performance is not guaranteed to be better. This might be explained by the fact that through taking the upper Hessenburg part of certain matrix component on the right side of (4.48) and (4.56) introduces instability to the track-

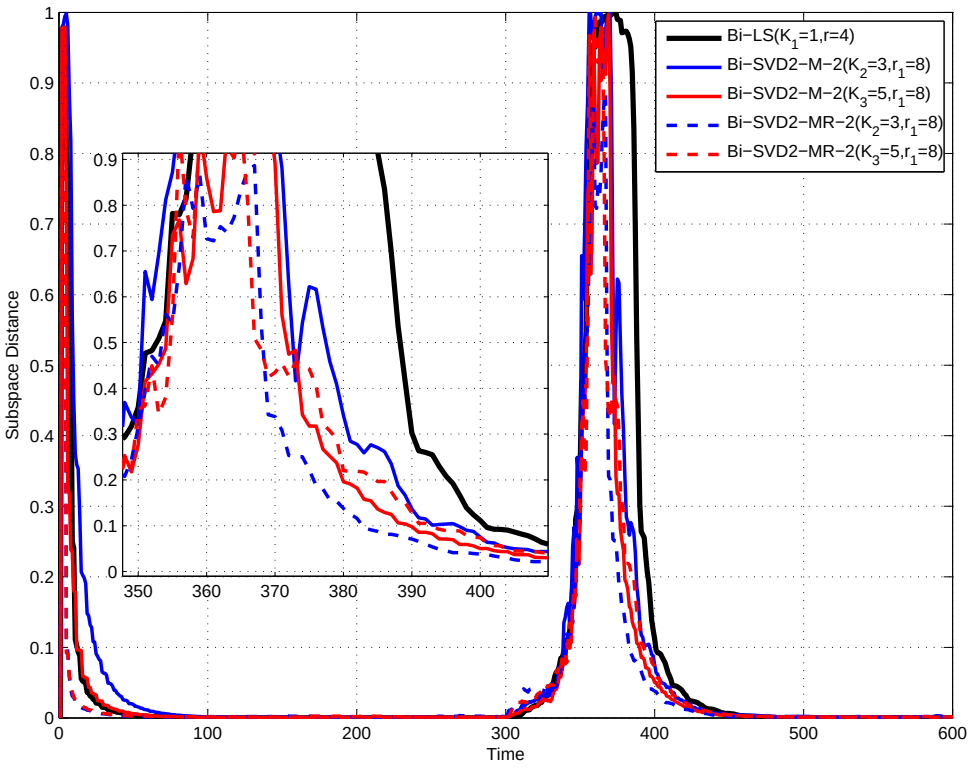

Figure 4.8: Bi-SVD2-M-2 and Bi-SVD2-MR-2 algorithms. $N = 80$, $L = 100$, $r = 4$, $r_1 = 8$, $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 100$, $SNR = 30$dB.

ing algorithms since it only happens at the last inner iterations as a sudden change for $\mathbf{Q}_B^{(K)}(t)$ and $\mathbf{Q}_A^{(K)}(t)$. However, when K is relatively small, Bi-SVD2-MR-2 algorithm still outperforms Bi-SVD2-M-2 for the same number of inner iterations as shown in Fig. 4.8.

4.6 Chapter Conclusion

In this chapter, we proposed new subspace tracking algorithms, Bi-SVD1-M-1 and Bi-SVD1-MR-1, Bi-SVD2-M-1 and Bi-SVD2-MR-1, by equipping bi-iteration SVD algo-

rithms with multiple inner iterations and Ritz acceleration using different consistent approximation data matrices. The necessity of tracking an enlarged subspace dimension is also explained and performance analysis is provided. Based on a more accurate data matrix approximation, algorithms with Ritz acceleration, Bi-SVD1-MR-1 and Bi-SVD2-MR-1, outperform Bi-SVD1-M-1, Bi-SVD2-M-1 and other existing tracking algorithms, showing excellent performance for both tracking accuracy and convergence rate as expected. Only a few number of inner iterations is enough for Bi-SVD1-MR-1 and Bi-SVD2-MR-1 to accomplish much improved performance so that the algorithm complexity is not much increased. When the tracking dimension becomes large, computational loads may be the main concern in practice and the faster version of previously proposed algorithms, Bi-SVD2-M-2 and Bi-SVD2-MR-2, are developed. Although the performance is not as good as higher complexity algorithms, the ultrafast versions do outperform the existing tracking algorithms.

Chapter 5

Conclusion

In this dissertation, we consider both optimal design of the source and relay matrices for a non-regenerative two-way MIMO relay system and fast subspace tracking algorithms for adaptive signal processing applications. For two-way relay system, an optimal structure for the relay matrix is shown in our study, which is useful for reducing the complexity of the optimal design when the relay has much more antennas than the users. The hybrid gradient method and the iterative WMMSE method are developed for relay matrix optimization and consistently produce the same results. While the hybrid gradient method is faster to converge, and the iterative WMMSE method is easier to implement. Compared to the SDP based optimization method in [77] which is only applicable to single-antenna users, both methods are much faster. For any given relay matrix, we demonstrate that the optimal design of the source matrices (for uniformly weighted sum rate) could be obtain by the generalized water filling algorithm in [76]. By alternating the relay optimization and the source optimization, our proposed joint source-and-relay optimization method can provide

much improved system capacity.

For thermal modeling and temperature prediction, a black-box LTI MIMO thermal model for multicore microprocessor system with power consumptions as inputs and temperature observations as outputs is adopted. We first apply LS method combined with model order selection rules such as AIC to estimate the model parameters and predict future temperatures. In order to incorporate the uncertainty inherited in model order selection, the idea of model averaging is introduced and, correspondingly, LS based algorithm with model screening is proposed whose model weights can be computed by S-AIC with improved prediction performance compared to LS method with AIC. In the future, model averaging algorithm could also be extended to online estimation of model parameters by Recursive Least Square (RLS) method.

In the third part, new fast subspace tracking algorithms are proposed through extending bi-iteration SVD algorithms with multiple inner iterations and Ritz acceleration. We demonstrate the necessity of tracking an enlarged subspace dimension and provide performance analysis for the new developed algorithms. Using different consistent approximation data matrices with larger rank, proposed algorithms without Ritz acceleration, Bi-SVD1-M-1, Bi-SVD2-M-1, could achieve better performance than existing algorithms. When further integrating Ritz acceleration into Bi-SVD1-MR-1 and Bi-SVD2-MR-1, those algorithms outperform Bi-SVD1-M-1 and Bi-SVD2-M-1, showing excellent performance for both tracking accuracy and convergence rate as expected. Since basically only the SVD of a much smaller-dimensional matrix is added at each inner iteration and a few number of inner iterations is enough for Bi-SVD1-MR-1 and Bi-SVD2-MR-1 to accomplish much

improved performance, the overall algorithm complexity is not much increased. We also proposed the faster version of previously proposed algorithms, Bi-SVD2-M-2 and Bi-SVD2-MR-2, for cases where the tracking dimension becomes large and then computational loads may be the main concern in practice. The ultrafast versions again outperform the existing tracking algorithms, although their performance is not as good as their higher complexity counterparts.

Bibliography

[1] K. Abed-Meraim, A. Chkeif, and Y. Hua. Fast orthonormal past algorithm. *IEEE Signal Processing Letters*, 7(3):60–62, Mar. 2000.

[2] Roland Badeau, Bertrand David, and Gaël Richard. Fast approximated power iteration subspace tracking. *IEEE Transactions on Signal Processing*, 53(8):2931–2941, Aug. 2005.

[3] Roland Badeau, Gaël Richard, and Bertrand David. Sliding window adaptive svd algorithms. *IEEE Transactions on Signal Processing*, 52(1):1–10, Jan. 2004.

[4] E. S. Baker and R. D. DeGroat. A correlation-based subspace tracking algorithm. *IEEE Transactions on Signal Processing*, 46(11):3112–3116, Nov. 1998.

[5] S. Bannour and M. R. Azimi-Sadjadi. An adaptive approach for optimal data reduction using recursive least squares learning method. *Proc of IEEE ICASSP*, Mar. 1992.

[6] Alireza Shahan Behbahani, Ricardo Merched, and Ahmed M. Eltawil. Optimizations of a MIMO relay network. *IEEE Transactions on Signal Processing*, 56:5062–5073, Dec. 2008.

[7] C. H. Bischof and G. M. Shroff. On updating signal subspaces. *IEEE Transactions on Signal Processing*, 40(1):96–105, Jan. 1992.

[8] Helmut Bölcskei, Rohit U. Nabar, Özgür Oyman, and Arogyaswami J. Paulraj. Capacity scaling laws in MIMO relay networks. *IEEE Transactions on Wireless Communications*, 5:1433–1444, Jun. 2006.

[9] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[10] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. *In Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, pages 171–182, 2001.

[11] Søren Skovgaard Christensen, Rajiv Agarwal, Elisabeth de Carvalho, and John M. Cioffi. Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design. *IEEE Transactions on Wireless Communications*, 7(12):4792–4799, Dec. 2008.

[12] M. Clint and A. Jennings. A simultaneous iteration method for the unsymmetric eigenvalue problem. *IMA Journal of Applied Mathematics*, 8(1):111–121, 1971.

[13] K. Coskun, T. S. Rosing, and K. C. Gross. Utilizing predictors for efficient thermal management in multiprocessor socs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1503–1516, Oct. 2009.

[14] Thomas M. Cover and Abbas A. El Gamal. Capacity theorems for the relay channel. *IEEE Transactions on Information Theory*, ITC25(5):572C584, Sep. 1979.

[15] R. D. DeGroat. Noniterative subsuace tracking. *IEEE Transactions on Signal Processing*, 40(3):571–577, Mar. 1992.

[16] Thom J. Eguia, Sheldon X.-D. Tan, Ruijing Shen, Eduardo H. Pacheco, and Murli Tirumala. General behavioral thermal modeling and characterization for multi-core microprocessor design. *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1136–1141, Mar. 2010.

[17] Zheng Fang, Yingbo Hua, and John C. Koshy. Joint source and relay optimization for a non-regenerative MIMO relay. *Proc. IEEE Workshop Sensor Array Multichannel Signal Processing*, pages 239–243, Jul. 2006.

[18] Gene H. Golub and Charles F. Van Loan, editors. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 3rd ed edition, 1996.

[19] M. Grant and S. Boyd. *CVX: Matlab software for disciplined convex programming*. http://stanford.edu/ boyd/cvx, 2008.

[20] Ingmar Hammerstrom, Marc Kuhn, Celal Esli, Jian Zhao, Armin Wittneben, and Gerhard Bauch. MIMO two-way relaying with transmit CSI at the relay. In *IEEE Signal Processing Advances in Wireless Communications, SPAWC 2007*, Helsinki, Finland, Jun. 2007. 5 pages.

[21] F. Hansen and P. Y. Yalamov. Computing symmetric rank-revealing decompositions via triangular factorization. *SIAM J. Matrix Analysis Appl.*, 23(2):443–458, 2001.

[22] Nils Lid Hjort and Gerda Claeskens. Frequentist model average estimators. *Journal of the American Statistical Association*, 98(464):879–899, Dec. 2003.

[23] Are Hjørungnes and David Gesbert. Complex-valued Matrix differentiation: Techniques and key results. *IEEE Transactions Signal Process.*, 55(6):2740–2746, Jun. 2007.

[24] Are Hjørungnes and David Gesbert. Hessians of scalar functions of complex-valued matrices: A systematic computational approach. *International Symposium on Signal Processing and Its Applications*, Dec. 2007.

[25] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[26] Anders Høst-Madsen and Junshan Zhang. Capacity bounds and power allocation for wireless relay channel. *IEEE Transactions on Information Theory*, 51(6):2020–2040, Jun. 2005.

[27] S. Hosur, A. H. Tewfik, and D. Boley. Ulv and generalized ulv subspace tracking adaptive algorithms. *IEEE Transactions on Signal Processing*, 46(5):1282–1297, May 1998.

[28] Y. Hua, A. Gershman, and Q. Cheng, editors. *High-Resolution and Robust Signal Processing*. Marcel Dekker, 2003.

[29] Y. Hua, Y. Xiang, T. Chen, K. Abed-Meraim, and Y. Miao. A new look at the power method for fast subspace tracking. *Digital Signal Processing*, 9(4):297–314, Oct. 1999.

[30] W. Huang, M. R. Stan, K. Sankaranarayanan K. Skadron, and S. Ghosh. Hotspot: A compact thermal modeling method for cmos vlsi systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, May. 2006.

[31] Ilkka Karasalo. Estimating the covariance matrix by signal subspace averaging. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(1):8–12, Jan. 1986.

[32] J. Karhunen. Adaptive algorithms for estimating eigenvectors of correlation type matrices. *Proc of IEEE ICASSP*, Mar. 1984.

[33] Steven M. Kay. *Fundamentals of Statistical Processing, Volume I: Estimation Theory*. Prentice Hall, 1993.

[34] Sang Joon Kim, Natasha Devroye, Patrick Mitran, and Vahid Tarokh. Achievable rate regions for bi-directional relaying. *IEEE Transactions on Information Theory*, submitted. available online.

[35] Sang Joon Kim, Natasha Devroye, Patrick Mitran, and Vahid Tarokh. Achievable rate regions for bi-directional relaying. IEEE Transactions on Information Theory. submitted.

[36] G. Kramer, M. Gastpar, and P. Gupta. Cooperative strategies and capacity theorems for relay networks. *IEEE Transactions on Information Theory*, 51(9):3037–3063, Sep. 2005.

[37] Namyoon Lee, Hyun Jong Yang, and Joohwan Chun. Achievable sum-rate maximizing af relay beamforming scheme in two-way relay channels. *IEEE International Conference on Communications Workshops*, May 2008.

[38] Duo Li, Sheldon X.-D. Tan, Eduardo Hernandez Pacheco, and Murli Tirumala. Architecture-level thermal characterization for multicore microprocessors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(10):495–1507, Oct. 2009.

[39] J. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, 1999.

[40] Y. Miao and Yingbo Hua. Fast subspace tracking and neural network learning by a novel information criterion. *IEEE Transactions on Signal Processing*, 46(7):1967–1979, Jul. 1998.

[41] M. Moonen, P. van Dooren, and J. Vandewalle. Updating singular value decompositions: A parallel implementation. *Proc. SPlE Adv. Algorithms Archetectures Signal Processing*, Aug. 1989.

[42] B. Muquet, M. de Courville, and P. Duhamel. Subspace-based blind and semi-blind channel estimation for ofdm systems. *IEEE Transactions on Signal Processing*, 50(7):1699–1712, Jul. 2002.

[43] E. Oja. A simplified neuron model as a principal component analyrer. *J . Math. Bio.*, 15:267–273, 1982.

[44] Shan Ouyang and Yingbo Hua. Bi-iterative least-square method for subspace tracking. *IEEE Transactions on Signal Processing*, 53(8):2984–2996, Aug. 2005.

[45] N. L. Owsley. Adaptive data orthogonalizaiton. *Proc of IEEE ICASSP*, 1978.

[46] Boris Rankov and Armin Wittneben. Achievable rate regions for the two-way relay channel. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 1668–1672, Seattle, Jul. 2006.

[47] Boris Rankov and Armin Wittneben. Spectral efficient protocols for half-duplex fading relay channels. *IEEE Journal on Selected Area In Communications*, 25(2):379–389, Feb 2007.

[48] Y. Rong and Y. Hua. Optimality of diagonalization of multi-hop MIMO relays. *IEEE Transactions on Wireless Communications*, 8(12):6068–6077, Dec 2009.

[49] Y. Rong, X. Tang, and Y. Hua. A unified framework for optimizing linear non-regenerative multicarrier MIMO relay communication systems. *IEEE Transactions on Signal Processing*, 57(12):4837–4852, Dec 2009.

[50] Andrew Sendonaris, Elza Erkip, and Behnaam Aazhang. User cooperation diversity-part i: System description. *IEEE Transactions on Communications*, 51(11):1927–1938, Nov. 2003.

[51] Andrew Sendonaris, Elza Erkip, and Behnaam Aazhang. User cooperation diversitypart ii: Implementation aspects and performance analysis. *IEEE Transactions on Communications*, 51(11):1939–1948, Nov. 2003.

[52] Andrew Sendonaris, Elza Erkip, and Behnaam Aazhang. User cooperation diversity-part i: System description. *IEEE Transactions on Information Theory*, 50(12):3062–3080, Dec. 2004.

[53] K. C. Shaman. Adaptive algorithms for estimating the complete covariance eifenstructure. *Proc of IEEE ICASSP*, Apr. 1986.

[54] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature aware microarchitecture. *Proc. Int. Symp. Comput. Architect.*, pages 2–13, Jun. 2003.

[55] G. W. Stewart. Methods of simultaneous iteration for calculating eigenvectors of matrices. *in Topics in Numerical Analysis II (ed. J. H. H. Miller), Academic Press, New York*, pages 185–196, 1975.

[56] G. W. Stewart. An updating algorithm for subspace tracking. *IEEE Transactions on Signal Processing*, 40(6):1535–1541, Jun. 1992.

[57] Petre Stoica and Yngve Selén. Model-order selection: a review of information criterion rules. *IEEE Signal Processing Magazine*, 21(4):36–47, Jul. 2004.

[58] Peter Strobach. Bi-iteration svd subspace tracking algorithms. *IEEE Transactions on Signal Processing*, 45(5):1222–1240, May 1997.

[59] Xiaojun Tang and Yingbo Hua. Optimal design of non-regenerative MIMO wireless relays. *IEEE Transactions on Wireless Communications*, 6:1398–1407, Apr. 2007.

[60] E. Telatar. Capacity of multi-antenna gaussian channels. *European transactions on telecommunications*, 10(6):585–595, 1999.

[61] D. W. Tufts and C. D. Melissinos. Simple, effective computation of principal eigenvectors and their eigenvalues and applications to highresolution estimation of frequencies. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1046–1053, Oct. 1986.

[62] Timo Unger and Anja Klein. Duplex schemes in multiple antenna two-hop relaying. *EURASIP Journal on Advances in Signal Processing*, 2008. Volume 2008, Article ID 128592, 14 pages.

[63] B. Egardt. V. U. Reddy and T. Kailath. Least squares type algorithm for adaptive implementation of pisarenko's harmonic retrieval method. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30(3):399–405, Jun. 1982.

[64] A. van den Bos. Complex gradient and hessian. *Proc. Inst. Elect. Eng. Vision, Image Signal Process.*, 6(141):380–383, Dec. 1994.

[65] Rahul Vaze and Robert W. Heath Jr. On the capacity and diversity-multiplexing tradeoff of the two-way relay channel. *IEEE Transactions on Information Theory.* to appear, available online.

[66] Bo Wang, Junshan Zhang, and Anders Høst-Madsen. On the capacity of mimo relay channels. *IEEE Transactions on Information Theory*, 51(1):29–43, Jan. 2005.

[67] X. Wang and H. V. Poor. Blind equalization and multiuser detection in dispersive cdma channels. *IEEE Transactions on Communications*, 46(1):91–103, Jan. 1998.

[68] T. K. Sarkar. X. Yang and E. Arvas. A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(10):1550–1556, Oct. 1989.

[69] Guanghan Xu and Thomas Kailath. Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE*, 78(8):1327–1343, Aug. 1990.

[70] Guanghan Xu and Thomas Kailath. Fast subspace decomposition. *IEEE Transactions on Signal Processing*, 42(3):539–551, Mar. 1994.

[71] Guanghan Xu, Hongyuan Zha, Gene H. Golub, and Thomas Kailath. Fast algorithms for updating signal subspaces. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41(8):537–549, Aug. 1994.

[72] Shengyang Xu and Yingbo Hua. Source-relay optimization for a two-way MIMO relay system. In *Proc of IEEE ICASSP'2010*, Dallas, TX, March 2010.

[73] Bin Yang. Projection approximation subspace tracking. *IEEE Transactions on Signal Processing*, 43(1):95–107, Jan. 1995.

[74] J. Yang and M. Kaveh. Adaptive eigensubspace algorithms for direction or frequency estimation and tracking. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(2):241–251, Feb. 1988.

[75] Yonghong Yang, Zhenyu Gu, Changyun Zhu, Robert P. Dick, and Li Shang. ISAC: Integrated space-and-time-adaptive chip-package thermal analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(1):86–99, 2007.

[76] Yuan Yu and Yingbo Hua. Power allocation for a MIMO relay system with multiple-antenna users. *IEEE Transactions on Signal Processing*, 58(5):2823–2835, May 2010.

[77] Rui Zhang, Ying-Chang Liang, Chin Choy Chai, and Shuguang Cui. Optimal beamforming for two-way multi-antenna relay channel with analogue network coding. *IEEE Journal on Selected Area In Communications*, 27(5):699–712, June 2009.

Appendix A

Appendices

A.1 Proof of Theorem 2.3.1

Without loss of generality, we can express $\mathbf{F}$ as

$$\mathbf{F} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_1^{\perp} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{U}_2^H \\ \mathbf{U}_2^{\perp H} \end{bmatrix} \tag{A.1}$$

where $\mathbf{V}_1^{\perp} \in \mathbb{C}^{M \times (M-2N)}$ is such that $\begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_1^{\perp} \end{bmatrix}$ is unitary, $\mathbf{U}_2^{\perp} \in \mathbb{C}^{M \times (M-2N)}$ is such that $\begin{bmatrix} \mathbf{U}_2 & \mathbf{U}_2^{\perp} \end{bmatrix}$ is unitary. In other words, for any given $\mathbf{F}$, there is a unique set of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$, and vice versa. Clearly, $\mathbf{V}_1^H \mathbf{V}_1^{\perp} = \mathbf{0}$ and $\mathbf{U}_2^{\perp H} \mathbf{U}_2 = \mathbf{0}$. Thus, from (2.9) and (2.10), we have

$$\begin{bmatrix} \mathbf{H}_{R1} \\ \mathbf{H}_{R2} \end{bmatrix} \mathbf{V}_1^{\perp} = \begin{bmatrix} \mathbf{H}_{R1} \mathbf{V}_1^{\perp} \\ \mathbf{H}_{R2} \mathbf{V}_1^{\perp} \end{bmatrix} = \mathbf{0} \tag{A.2}$$

$$\mathbf{U}_2^{\perp H}\begin{bmatrix}\mathbf{H}_{1R} & \mathbf{H}_{2R}\end{bmatrix} = \begin{bmatrix}\mathbf{U}_2^{\perp H}\mathbf{H}_{1R} & \mathbf{U}_2^{\perp H}\mathbf{H}_{2R}\end{bmatrix} = \mathbf{0} \tag{A.3}$$

Using the above properties of $\mathbf{F}$ in (2.5), we have

$$\begin{aligned}\mathbf{X}_i = \mathbf{I} + (\mathbf{H}_{Ri}\mathbf{V}_1\mathbf{A}\mathbf{A}^H\mathbf{V}_1^H\mathbf{H}_{Ri}^H + \mathbf{H}_{Ri}\mathbf{V}_1\mathbf{B}\mathbf{B}^H\mathbf{V}_1^H\mathbf{H}_{Ri}^H + \mathbf{I})^{-1}\cdot \\ \mathbf{H}_{Ri}\mathbf{V}_1\mathbf{A}\mathbf{U}_2^H\mathbf{H}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{H}_{\bar{i}R}^H\mathbf{U}_2\mathbf{A}^H\mathbf{V}_1^H\mathbf{H}_{Ri}^H\end{aligned} \tag{A.4}$$

We see that the matrices $\mathbf{C}$ and $\mathbf{D}$ do not affect $\mathbf{X}_i$ and hence the rates r_i. We now apply the properties of $\mathbf{F}$ to (2.2). We then have

$$\begin{aligned}p_R = &\operatorname{Tr}(\mathbf{A}\mathbf{U}_2^H\mathbf{H}_{1R}\mathbf{R}_{x_1}\mathbf{H}_{1R}^H\mathbf{U}_2\mathbf{A}^H) \\ &+ \operatorname{Tr}(\mathbf{C}\mathbf{U}_2^H\mathbf{H}_{1R}\mathbf{R}_{x_1}\mathbf{H}_{1R}^H\mathbf{U}_2\mathbf{C}^H) \\ &+ \operatorname{Tr}(\mathbf{A}\mathbf{U}_2^H\mathbf{H}_{2R}\mathbf{R}_{x_2}\mathbf{H}_{2R}^H\mathbf{U}_2\mathbf{A}^H) \\ &+ \operatorname{Tr}(\mathbf{C}\mathbf{U}_2^H\mathbf{H}_{2R}\mathbf{R}_{x_2}\mathbf{H}_{2R}^H\mathbf{U}_2\mathbf{C}^H) + \operatorname{Tr}(\mathbf{A}^H\mathbf{A}) \\ &+ \operatorname{Tr}(\mathbf{B}^H\mathbf{B}) + \operatorname{Tr}(\mathbf{C}^H\mathbf{C}) + \operatorname{Tr}(\mathbf{D}^H\mathbf{D})\end{aligned} \tag{A.5}$$

which shows that $\mathbf{C}$ and $\mathbf{D}$ only increase the transmit power at the relay unless they are zero. Therefore, the optimal choice of $\mathbf{C}$ and $\mathbf{D}$ is simply $\mathbf{C} = 0$ and $\mathbf{D} = 0$. So, we can now write

$$\begin{aligned}p_R = &\operatorname{Tr}(\mathbf{A}\mathbf{U}_2^H\mathbf{H}_{1R}\mathbf{R}_{x_1}\mathbf{H}_{1R}^H\mathbf{U}_2\mathbf{A}^H) + \operatorname{Tr}(\mathbf{A}^H\mathbf{A}) \\ &+\operatorname{Tr}(\mathbf{A}\mathbf{U}_2^H\mathbf{H}_{2R}\mathbf{R}_{x_2}\mathbf{H}_{2R}^H\mathbf{U}_2\mathbf{A}^H) + \operatorname{Tr}(\mathbf{B}^H\mathbf{B})\end{aligned} \tag{A.6}$$

The remaining task is to prove that the optimal $\mathbf{B}$ is also zero. Define the positive semidefinite matrices

$$\mathbf{J}_i(\mathbf{X}) = \mathbf{H}_{Ri}\mathbf{V}_1\mathbf{X}\mathbf{X}^H\mathbf{V}_1^H\mathbf{H}_{Ri}^H$$

$$\mathbf{L}_i(\mathbf{X}) = \mathbf{H}_{Ri}\mathbf{V}_1\mathbf{X}\mathbf{U}_2^H\mathbf{H}_{\bar{i}R}\mathbf{R}_{x_{\bar{i}}}\mathbf{H}_{\bar{i}R}^H\mathbf{U}_2\mathbf{X}^H\mathbf{V}_1^H\mathbf{H}_{Ri}^H$$

Then, $r_i = \frac{1}{2}\log_2\det(\mathbf{X}_i)$ becomes

$$r_i = \frac{1}{2}\log_2\det\left(\mathbf{I} + (\mathbf{J}_i(\mathbf{A}) + \mathbf{J}_i(\mathbf{B}) + \mathbf{I})^{-1}\mathbf{L}_i(\mathbf{A})\right) \tag{A.7}$$

By Lemma 1 shown later, (A.7) implies

$$r_i \leq \frac{1}{2}\log_2\det\left(\mathbf{I} + (\mathbf{J}_i(\mathbf{A}) + \mathbf{I})^{-1}\mathbf{L}_i(\mathbf{A})\right) \tag{A.8}$$

where the upper bound is achieved when $\mathbf{B} = 0$. We also know that $\mathbf{B}$ only increases p_R unless $\mathbf{B} = 0$. Therefore, the optimal $\mathbf{B}$ is zero.

Lemma 1: Given the conjugate symmetric matrices $\mathbf{X} \geq 0$, $\mathbf{Y} > 0$ and $\mathbf{Z} \geq 0$, it follows that

$$\det\left(\mathbf{I} + (\mathbf{Y} + \mathbf{Z})^{-1}\mathbf{X}\right) \leq \det(\mathbf{I} + \mathbf{Y}^{-1}\mathbf{X})$$

Proof. We can write $\det\left(\mathbf{I}+(\mathbf{Y}+\mathbf{Z})^{-1}\mathbf{X}\right) = \det\left(\mathbf{I}+\mathbf{X}^{\frac{1}{2}}\mathbf{Y}^{-\frac{1}{2}}(\mathbf{I}+\mathbf{Y}^{-\frac{H}{2}}\mathbf{Z}\mathbf{Y}^{-\frac{1}{2}})^{-1}\mathbf{Y}^{-\frac{H}{2}}\mathbf{X}^{\frac{H}{2}}\right)$ and $\det(\mathbf{I} + \mathbf{Y}^{-1}\mathbf{X}) = \det\left(\mathbf{I} + \mathbf{X}^{\frac{1}{2}}\mathbf{Y}^{-\frac{1}{2}}\mathbf{Y}^{-\frac{H}{2}}\mathbf{X}^{\frac{H}{2}}\right)$. We know $\mathbf{I} + \mathbf{Y}^{-\frac{H}{2}}\mathbf{Z}\mathbf{Y}^{-\frac{1}{2}} \geq \mathbf{I}$, $(\mathbf{I} + \mathbf{Y}^{-\frac{H}{2}}\mathbf{Z}\mathbf{Y}^{-\frac{1}{2}})^{-1} \leq \mathbf{I}$ and $\mathbf{X}^{\frac{1}{2}}\mathbf{Y}^{-\frac{1}{2}}(\mathbf{I} + \mathbf{Y}^{-\frac{H}{2}}\mathbf{Z}\mathbf{Y}^{-\frac{1}{2}})^{-1}\mathbf{Y}^{-\frac{H}{2}}\mathbf{X}^{\frac{H}{2}} \leq \mathbf{X}^{\frac{1}{2}}\mathbf{Y}^{-\frac{1}{2}}\mathbf{Y}^{-\frac{H}{2}}\mathbf{X}^{\frac{H}{2}}$. Since $\mathbf{A}' \leq \mathbf{B}'$ implies $\det(\mathbf{A}') \leq \det(\mathbf{B}')$, therefore $\det\left(\mathbf{I} + (\mathbf{Y} + \mathbf{Z})^{-1}\mathbf{X}\right) \leq \det(\mathbf{I} + \mathbf{Y}^{-1}\mathbf{X})$. □

A.2 Zoomed SDP Algorithm

In [77], a special case of the two-way MIMO relay system was considered where the two users are each with a single antenna. For this case, there is no need for source optimization since $\mathbf{R}_{x_1}$ and $\mathbf{R}_{x_2}$ are simply P_1 and P_2 respectively. And to maximize the sum rate $r_1 + r_2$, the relay optimization problem is reduced to

$$\begin{aligned} \max_{\mathbf{F}} \quad & \frac{1}{2}\log_2\left(1+\frac{|\mathbf{h}_{R1}^T\mathbf{F}\mathbf{h}_{2R}|^2P_2}{||\mathbf{F}^H\mathbf{h}_{R1}^*||^2+1}\right) \\ & \qquad +\frac{1}{2}\log_2\left(1+\frac{|\mathbf{h}_{R2}^T\mathbf{F}\mathbf{h}_{1R}|^2P_1}{||\mathbf{F}^H\mathbf{h}_{R2}^*||^2+1}\right) \\ s.t. \quad & ||\mathbf{F}\mathbf{h}_{1R}||^2P_1 + ||\mathbf{F}\mathbf{h}_{2R}||^2P_2 + \mathrm{Tr}(\mathbf{F}\mathbf{F}^H) \leq P_R \end{aligned} \tag{A.9}$$

Since each user has a single antenna, the channel matrices $\mathbf{H}_{1R}$, $\mathbf{H}_{R1}$, $\mathbf{H}_{2R}$ and $\mathbf{H}_{R2}$, are now reduced to channel vectors $\mathbf{h}_{1R}$, $\mathbf{h}_{R1}$, $\mathbf{h}_{2R}$ and $\mathbf{h}_{R2}$. This problem is still non-convex. In order to use a convex optimization program to solve this problem, the authors of [77] proposed to fix the ratio between r_1 and r_2. Specifically, they introduced the rate profile vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]$ where $\alpha_1 = \frac{r_1}{r_{sum}}$, $\alpha_2 = 1 - \alpha_1 = \frac{r_2}{r_{sum}}$ and $r_{sum} = r_1 + r_2$. For each fixed $\boldsymbol{\alpha}$, the sum rate r_{sum} is maximized by solving the following problem

$$\begin{aligned} \max_{\mathbf{F}} \quad & r_{sum} \\ s.t. \quad & \frac{1}{2}\log_2\left(1+\frac{|\mathbf{h}_{R1}^T\mathbf{F}\mathbf{h}_{2R}|^2P_2}{||\mathbf{F}^H\mathbf{h}_{R1}^*||^2+1}\right) \geq \alpha_1 r_{sum} \\ & \frac{1}{2}\log_2\left(1+\frac{|\mathbf{h}_{R2}^T\mathbf{F}\mathbf{h}_{1R}|^2P_1}{||\mathbf{F}^H\mathbf{h}_{R2}^*||^2+1}\right) \geq \alpha_2 r_{sum} \\ & ||\mathbf{F}\mathbf{h}_{1R}||^2P_1 + ||\mathbf{F}\mathbf{h}_{2R}||^2P_2 + \mathrm{Tr}(\mathbf{F}\mathbf{F}^H) \leq P_R \end{aligned} \tag{A.10}$$

which in turn can be solved by a general purpose SDP algorithm such as in CVX [19]. For more details, see [77] .

To find the maximum sum rate, we have to solve the additional problem $\max_{0\le\alpha_1\le 1} r_{sum}$. One method is the brute force search within $0 \le \alpha_1 \le 1$, which is too costly. To do it more efficiently, we formulate a zoomed SDP algorithm:

Algorithm A.1 (zoomed SDP)**.**

1: ***Initialization:*** *Let L be an even integer larger than or equal to 4. Choose a small number ϵ. Partition $[0,1]$ into L uniform segments each of length $\delta = 1/L$, which yields $L-1$ interior uniform sample points $0 < \alpha_1^{(1)}, \cdots, \alpha_1^{(L-1)} < 1$.*

2: ***repeat***

3: *Run the above SDP algorithm to compute the maximum of r_{sum} for each of the $L-1$ sample points.*

4: *Determine the best sample point: $\alpha_1^* = \arg\max_{\alpha_1^{(1)},\cdots,\alpha_1^{(L-1)}} r_{sum}$.*

5: *Partition $[\alpha_1^* - \delta, \alpha_1^* + \delta]$ into L uniform segments (i.e., "zooming"), which resets the $L-1$ uniform sample points $\alpha_1^{(1)}, \cdots, \alpha_1^{(L-1)}$. Also reset $\delta := 2\delta/L$*

6: ***until*** $\delta < \epsilon$

By choosing L to be an even integer, we ensure that each new set of $L-1$ sample points include α_1^* in Step 4. We also need $L = 2m$ with $m \ge 2$. Otherwise, if $L = 2$, δ would stay the same and the algorithm would not work. The zoomed SDP algorithm is used to compare with the hybrid gradient algorithm and the iterative WMMSE algorithm developed in this dissertation. We choose $\epsilon = 10^{-3}$ and $L = 4$ in simulations.