USC-SIPI REPORT #156

Adaptive Linearly-Constrained Filtering: Principles and Implementations

by

Ching-Yih Tseng

May 1990

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

Dedication

This dissertation is dedicated to my parents.

Acknowledgements

I would like to thank Professor Lloyd J. Griffiths for his gracious guidance and constant support which made the completion of this dissertation possible. I would also like to express my gratitude to Professors Rama Chellappa, Richard Leahy and Chunming Wang for their participation in my Dissertation Committee.

My experience in USC would not have been so wonderful and exciting without many unforgettable friendships. I especially appreciate the opportunity to work with David Feldman, Michael Rude and George Zunich in the same research group. I acknowledge their patience in proof-reading my writing and their willingness in helping my various English problems. Because of their good nature, working with them has been so enjoyable. I also appreciate the opportunity to befriend many other SIPI members.

I am deeply indebted to my parents for their love and encouragement. This dissertation is dedicated to them.

This work was supported by the Semiconductor Research Cooperation under Contract No. 86-01-075 and No. 88-DP-075.

Contents

List of Tables

List of Figures

Abstract

Incorporating linear constraints in adaptive filters has arisen as a new trend to provide robust filtering operations in signal processing. The mechanism of this class of adaptive filter is to minimize the output power while constraining the weight values by a set of linear equations. The constraints are deliberately selected to confine the weights so that the signals of interest will not be cancelled when minimizing the filter output power. Through this constrained power minimization process, the noise power is reduced without distorting the signals of interest and therefore it improves the signal to noise power ratio at the output.

The basic requirement to utilize the adaptive linearly-constrained filter is to design the constraints. Appropriately setting up constraints requires the knowledge of the features regarding the particular signal of interest. Such features can be identified either through empirical methods, such as using a large set of training signals, or through theoretical methods, such as using the statistical information about the signals. After all, the resulting constraints have to effectively prevent the cancellation of signals of interest when the filter is applied. The second requirement is an adaptive algorithm to update the weights. To ensure implementation efficiency, it is desirable to develop implementation structures which are both computationally efficient and numerically stable.

This dissertation presents some results from the study of the above two requirements in adaptive linearly-constrained filters. To develop procedures for constraint specification and performance evaluation, the understanding of the effect of the constraints on the steady-state behaviors of the filter is an essential. Such effect is studied here through examining the optimal weights, the signal-to-noise power ratios, and the addition of constraints. On the other hand, a model for implementation is established

based on decoupling the weights into fixed and adaptive portions. By this decoupling, any unconstrained adaptive algorithm can be used to implement the adaptive-weight portion. Thus, only the fixed-weight implementation is studied in detailed in which the addressing issues include implementation structures and quantization effects. In the final part of this dissertation, a generalization of the adaptive linearly-constrained filters which allows the constraints to be adjustable is also discussed.

Chapter 1

Introduction

1.1 Motivation

Today's communication systems are operated in an increasingly noisy environment due to both growing amounts of traffic and the increased use of wide bandwidth encoding systems. As a direct result of spectral crowding, modern system must combat relatively high levels of co-channel interferences. One particularly effective technique which has been developed for reducing the effects of interfering signals is the use of adaptive filtering methods. Adaptive filters are able to provide more robust filtering operations than fixed weight filters in a time-varying environment due to their inherent ability to adapt to changes in the weighting values within the filter.

Traditional adaptive algorithms such as Recursive Least-Squares (RLS) and Least-Mean Square (LMS) approaches require the presence of a desired signal to the filter in which the filter weights are adjusted iteratively so that the filter output is a least-squares estimate of the desired signal. In [1, 2], several methods have been presented to select desired signals in various applications. In general, selecting a desired signal in the conventional adaptive filter for a particular application is always a difficult problem and therefore inevitably places a limitation in the applicability of the filter. To overcome this limitation, Griffiths proposed a simple adaptive algorithm in [3]. This algorithm is a modification of the LMS algorithm and is also known as the P-vector algorithm. A significant advantage of the P-vector algorithm is that it

has the same simplicity as the LMS algorithm while it does not require a desired signal. Instead, it utilizes the information about the cross-correlation between the desired signal and the filter input for updating the filter weights. It was shown in [3] that this cross-correlation information is usually known *a priori* or can be estimated easily in the application of antenna arrays.

A more recent technique to combat with the desired signal problem in adaptive filter is to incorporate linear constraints in the filter [4]. This technique has received significant interest in the signal processing and communication communities due to its wide applicability in the general area of wide bandwidth communications. In this dissertation, this particular class of adaptive filter is termed the *adaptive linearly-constrained filter*. In general, an adaptive linearly-constrained filter can be formulated as iteratively searching the optimal weights which minimize the filter output power subject to a set of linear constraints. The linear constraints are pre-selected to confine the weights so that the output power contributed by the desired signal retains less distortion than that of the noise when the filter weights converge. By this mean, the ratio of signal to noise power is improved via the filter. Clearly, some *a priori* knowledge must be available regarding the feature of the desired signal. Otherwise, there is no means of identifying the desired signal. Comparatively, obtaining this *a priori* knowledge is usually easier than getting access to the desired signal itself in most application.

The linear constraint method is more general than the P-vector method since more than the cross-correlation information can be incorporated in the filter. In fact, it was shown in [5] that the P-vector algorithm can be alternatively implemented by an adaptive linearly-constrained filter with one single constraint. However, using multiple constraints is far more robust and flexible than using single constraint in solving adaptive problems when a desired signal is not available.

The work in this dissertation is motivated to provide an integrated investigation of the adaptive linearly-constrained filter. However, it would be impossible to cover all important issues associated with the adaptive linearly-constrained filter in a dissertation of reasonable size, hence only those which serve as fundamental principles

toward understanding filter performances and those which have potential extension toward future applications are selected in this dissertation. In accordance with this philosophy, this dissertation is organized under two major topics: *Principles* and *Implementations*. By understanding the principle part one should be able to obtain some guidelines to design constraints and to evaluate the filter performance for his/her interested application, and by understanding the implementation part one should be able to realize his/her system with an optimal trade-off between complexity and performance.

1.2 Adaptive Filtering

This section provides an overview of the development of adaptive filtering starting from adaptive filters with desired signals, to those with linear constraints, and finally, to those with time-varying linear constraints. We show that these three evolutionary steps represent a natural extension from one to the next when viewed from linearly-constrained minimization standpoint.

1.2.1 Adaptive filters with desired signals

A typical model for representing an adaptive filter with desired signal is shown in Figure 1.1. The input vector $\mathbf{x}(n)$ linear combines with the weight vector $\mathbf{w}(n)$ to produce an output signal $y(n)$ which provides a least-squares estimate of the desired signal $d(n)$; the error signal $e(n)$ is formed by the difference of $d(n)$ and $y(n)$ and is used to adapt the filter weights toward optimality. Generally, $\mathbf{x}(n)$ is modeled as a stationary stochastic vector with zero mean and covariance matrix $\mathbf{R}_{xx}$. As a least-mean-squares estimator of $d(n)$, the filter reaches its optimality when the mean value of $\mathbf{w}(n)$ reaches the Weiner solution which is given by

$$\mathbf{w}_{weiner} = \mathbf{R}_{xx}^{-1}\mathbf{P}_{xd} \tag{1.1}$$

where $\mathbf{P}_{xd}$ is the cross correlation between $\mathbf{x}(n)$ and $d(n)$.

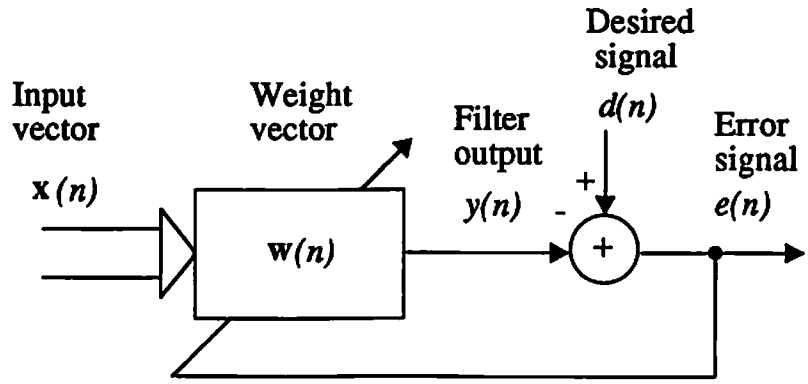

Figure 1.1: A representation model for adaptive filter with desired signal.

This conventional model has been widely used in the literature for analyzing adaptive filtering [6, 7]. The analysis is usually preformed from a vector space point of view in which the filter weights at any time instance is considered as a vector in a N-dimensional vector space where N is the size of the system. The update of the weights then corresponds to the movement of one vector to another in the vector space such that the magnitude of the error signal is reduced in a mean-square sense. A particularly simple algorithm for adapting the weights is the well-known LMS approach [8] which is based on a stochastic gradient search in the vector space.

To relate the conventional adaptive filter to a linearly-constrained problem, we can view the desired signal $d(n)$ as part of the input to the filter and increase correspondingly the dimension of the filter weight by one. To this end, we concatenate $d(n)$ and $\mathbf{x}(n)$ to form a new input vector to the filter,

$$\tilde{\mathbf{x}}(n) = \begin{bmatrix} d(n) \\ \mathbf{x}(n) \end{bmatrix} \tag{1.2}$$

and denote the new weight as

$$\tilde{\mathbf{w}}(n) = \begin{bmatrix} w_0 \\ -\mathbf{w}(n) \end{bmatrix}. \tag{1.3}$$

Clearly, to keep the filter unchanged by this reformulation we need to restrict the first component of the new weight to be unity. This can be done by restricting $\tilde{\mathbf{w}}$ to

satisfy the following equation,

$$\mathbf{c}^{\dagger}\tilde{\mathbf{w}} = 1, \tag{1.4}$$

where

$$\mathbf{c} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1.5}$$

and the superscript † denotes the Hermitian transpose. Under this constraint, the error signal $e(n)$ is given by

$$e(n) = \tilde{\mathbf{w}}^{\dagger}(n)\tilde{\mathbf{x}}(n). \tag{1.6}$$

As a result, we can formulate the conventional adaptive filter as a linearly-constrained minimum power problem defined by iteratively finding $\tilde{\mathbf{w}}$ which minimizes

$$\tilde{\mathbf{w}}^{\dagger}\tilde{\mathbf{R}}_{xx}\tilde{\mathbf{w}} \tag{1.7}$$

subject to the linear constraint given in (1.4), where $\tilde{\mathbf{R}}_{xx}$ is the covariance matrix of $\tilde{\mathbf{x}}$.

1.2.2 Adaptive filters with linear constraints

As mentioned earlier that by formulating the desired signal as one component of the input vector to the filter we can use a simple constraint to retrieve the desired signal from the filter input. Such a simple constraint is beneficial, however, only when the desired signal appears explicitly in the filter. Otherwise, the constraint is too simple to be useful in most applications. An extension of the conventional adaptive filter is therefore needed to cope with problems in which the desired signal is not explicitly presented. Under these circumstances, we need to incorporate more elaborated constraint to extract the desired signal from the input vector.

The first example of using more intricated constraint to avoid the need of desired signal was the P-vector algorithm presented in [3]. Instead of using a desired signal, the P-vector algorithm utilizes the cross-correlation information between the desired

signal and the input vector for weight adaptation. Although not noted in the original work, it was shown in [5] that the P-vector algorithm in fact implements a linearly-constrained minimum power problem in which the cross-correlation vector between the input vector and the desired signal is used as the constraint vector. This algorithm has been found useful in extracting narrowband signal in broadband noise background in which the cross-correlation information is known or can be estimated easily.

In some applications, however, single constraint may not be sufficient to extract the desired signal. This suggests a generalization to the linearly-constrained minimum power problem with multiple constraints. We define this problem as

$$\min_{\mathbf{w}} \ \mathbf{w}^{\dagger}\mathbf{R}_{xx}\mathbf{w} \tag{1.8}$$

subject to M linear constraints,

$$\mathbf{C}^{\dagger}\mathbf{w} = \mathbf{f}. \tag{1.9}$$

The *constraint matrix* $\mathbf{C}$ contains M column vectors while the *response vector* $\mathbf{f}$ specifies the corresponding constraint value for each vector. Without loss of generality, the matrix $\mathbf{C}$ is assumed full rank. A filter which iteratively solves this constrained problem is termed the *adaptive linearly-constrained filter.*

The application of adaptive linearly-constrained filters has been widely studied in the area of adaptive beamforming in array signal processing [9, 10, 11, 12, 13]. These studies can be generally divided into two major concerns, one is the principle of incorporating the constraint and the other is the implementation of the filter. In the previous work, several approaches have been proposed to construct the constraints which require only that the direction of arrival and a frequency band of interest be specified *a priori.*

This dissertation pursues studies in both the principle and implementation of adaptive linearly-constrained filters with the effort of providing general guidelines and leading to new applications in this topic. We analyze some important aspects of the steady-state behavior of the linearly-constrained filter, including the optimal weights, the signal-to-noise ratios, and the addition of constraints. The understanding of these properties is important since it helps evaluate the steady-state performance

for a particular application. Transient-state properties such as convergence rate and misadjustment are not discussed in this dissertation since, as will become clear in later chapters, they are basically the same as those of the conventional adaptive filter for which these properties have already been well-studied in the literature [6, 14, 15, 7, 16]. We also compare three iterative algorithms for solving the linearly-constrained minimum power problem with multiple constraints as defined earlier. Efficient structures based on the simplest iterative algorithm of the three are then derived to implement adaptive linearly-constrained filters with arbitrarily given constraints. Quantization studies on these structures are also given.

1.2.3 Adaptive filter with adjustable linear constraints

A remaining issue studied in this dissertation is to generalize the linearly constrained filter such that the righthand sides of the constraint equations are adjustable. This corresponds to incorporating a time variable n in the response vector $\mathbf{f}$ in the specification of constraints:

$$\mathbf{C}^{\dagger}\mathbf{w} = \mathbf{f}(n). \tag{1.10}$$

An adaptive filter with this type of time-varying constraints is termed as having *adjustable linear constraints.* An efficient structure is presented in the dissertation to implement this type of adaptive filter. Further study is required to exploit possible applications of adaptive filters with adjustable constraints.

1.3 Dissertation Outline

In Chapter 2 of this dissertation, the adaptive linearly-constrained filter is formulated as a minimization problem with linear constraints. Examples are given to illustrate the selection of constraints for both narrow and broad band signals. The steady-state performances of adaptive linearly-constrained filters including the optimal weights, the signal-to-noise ratios, and the addition of constraints are studied in detail.

In Chapter 3, three iterative algorithms for implementing the adaptive linearly-constrained filter are compared. This comparison is motivated to provide a clear

insight into the algorithms from both algebraic and geometric standpoints. It is shown that under certain conditions these three algorithms are identical in both transient and steady states. Two models, one useful for implementation while the other useful for analysis, are then established to represent the adaptive linearly-constrained filter.

In Chapter 4, two structures are derived to implement the adaptive linearly-constrained filter. The option of using one of these two structures guarantees implementation efficiency for a filter with either small or large number of constraints. A VLSI systolic array architecture is also presented to implement these structures.

In Chapter 5, quantization issues of implementing the adaptive linearly constrained filter in fixed-point are addressed. Simulation results are used to demonstrate the effect of quantization noise using both random and deterministic constraint matrices.

In Chapter 6, the implementation of adaptive filters with adjustable constraints is addressed. This implementation is an extension of the implementation structures in Chapter 4.

In Chapter 7, the dissertation is summaried and future related research topics are outlined.

1.4 Contributions

The main contributions of this dissertation are outlined in the following.

1) Derive closed-form expressions for the optimal weight of a linearly-constrained filter based on the square-root decomposition of the data covariance matrix. These expressions provide valuable insight to algorithm development in linearly-constrained filters.

2) Derive useful lower and upper limits for the optimal output signal-to-noise (SNR) ratio of a linearly-constrained filter. Define two factors which measure the ratios of the actual SNR to its lower and upper limits, respectively. It is shown that these two factors can be bounded in terms of the jammer-to-white noise ratio (JNR) at the input and several other places in the filter. By estimating the JNR, the bounds

on the two factors can be used to predict the performance of the adaptive linearly-constrained filter in a particular noise environment.

3) Derive recursive formulas to update the optimal weight and output power expression when an additional single constraint is added to the linearly-constrained filter. These formulas clearly indicate the effect of adding constraints to the filter.

4) Provide a unified view of three iterative algorithms used to implement the adaptive linearly-constrained filter. Conditions under which these three algorithms are identical are established.

5) Develop two structures to implement the adaptive linearly-constrained filter with small and large number of constraints, respectively. Various transformations such as Gauss transformation, Householder transformation and Givens rotation, can be used in the two structures to obtain computationally efficient and numerically stable implementations.

6) Establish several performance measures for studying the quantization noise in the adaptive linearly-constrained filter. Introduce a simple arithmetic — Power-of-Two — to reduce the hardware implementation cost of an adaptive linearly-constrained filter while maintaining acceptable quantization error.

7) Propose a generalization of the adaptive linearly-constrained filter by allowing the linear constraints to be time-varying. An efficient structure is also derived for implementation.

Chapter 2

Linearly-Constrained Filter

2.1 Introduction

The linearly-constrained filter is referred to here as a FIR filter with its weights being constrained to satisfy one or more linear equations while minimizing the filter output power. Such filters have been widely used as optimal receivers in array signal processing [9, 10, 11, 12, 13]. With the purpose of providing some guidelines for its performance analysis and extending the application to the general area of digital signal processing, this chapter studies the performance of the linearly-constrained filter and some of its interesting properties. The study is only conducted with the filter in steady-state even though linearly-constrained filters are practically implemented in an adaptive manner. Transient-state properties such as convergence rate and misadjustment are not discussed here since, as will become clear later in the thesis, they are basically the same as those of the conventional adaptive filter for which these properties have already been well-studied in the literature [6, 14, 15, 7, 16].

In this chapter, the analysis of the linearly-constrained filter is mainly based on the use of linear algebra techniques. The linearly-constrained filter is first formulated as a minimization problem with linear constraints in Section 2.2. In Section 2.3, we present two simple examples to illustrate the idea of constraint specifications for both narrow band and broad band signals. Three subsequent sections are respectively devoted to studying the optimal weights, the signal-to-noise ratios, and the addition

of constraints in the linearly-constrained filter. In the summary section, we make some remarks concerning the use of results obtained in this chapter.

2.2 Problem Formulation

2.2.1 Multi-channel FIR Filter

Figure 2.1 illustrates a direct form multi-channel FIR filter with K sensors and L tapped-delay lines. At each snapshot, the data appearing at the K sensors form a K-dimensional data vector designated as

$$\chi(n) = \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_K(n) \end{bmatrix}. \tag{2.1}$$

This vector is termed the *filter snapshot vector.* Since each sensor has a tapped delay-line of length L, we can form a KL-dimensional vector termed the *stacked snapshot vector.* This is denoted by

$$\mathbf{x}(n) = \begin{bmatrix} \chi(n) \\ \chi(n-1) \\ \vdots \\ \chi(n-L+1) \end{bmatrix}. \tag{2.2}$$

In this dissertation, the data are assumed to be complex-valued for generality. The complex data may have resulted from the use of in-phase/quadrature-phase modulation. The output of the filter is given by the inner product of the stacked snapshot vector and the KL-dimensional complex valued weight vector $\mathbf{w}(n)$, that is,

$$y(n) = \mathbf{w}^{\dagger}(n)\mathbf{x}(n) \tag{2.3}$$

where the notation † denotes the Hermitian transpose. Note that it is the complex conjugate of $\mathbf{w}(n)$ rather than $\mathbf{w}(n)$ which operates on the data. Denoting the weight

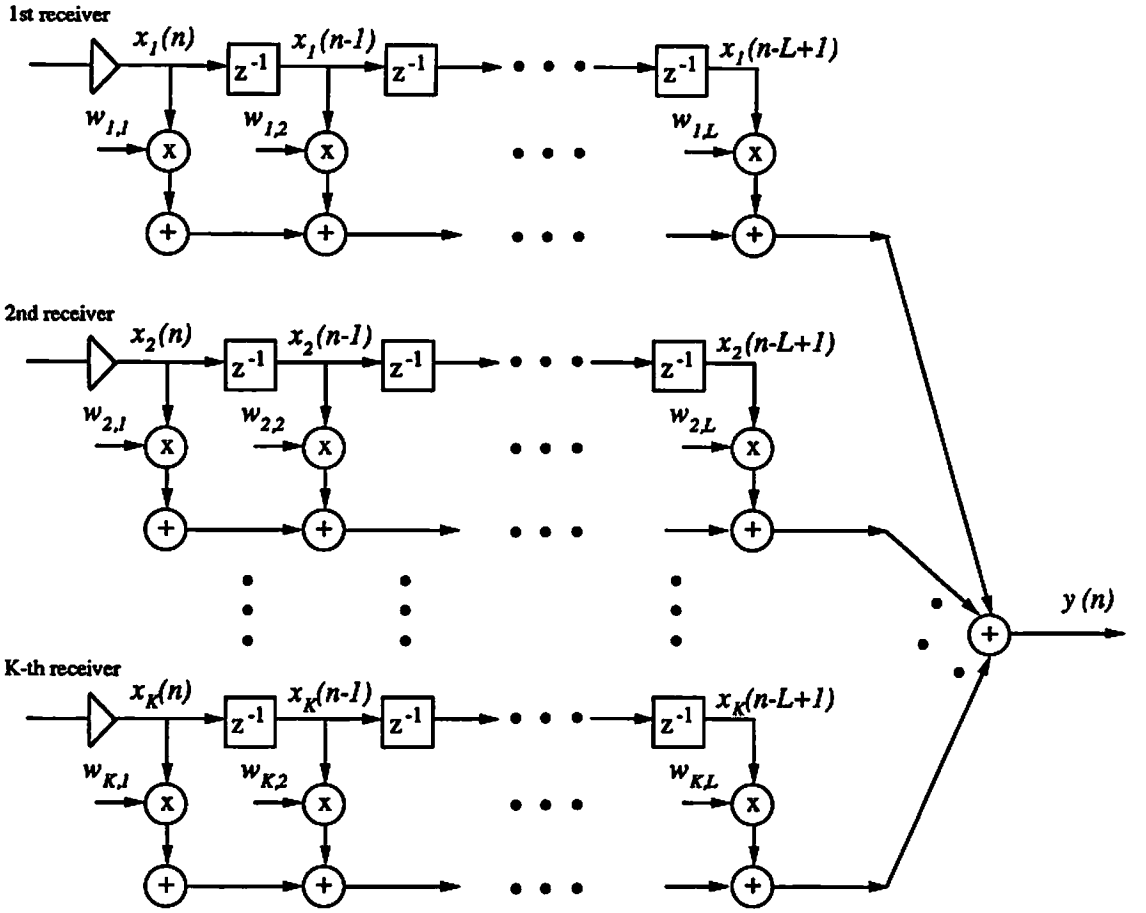

Figure 2.1: The direct form multi-channel FIR filter.

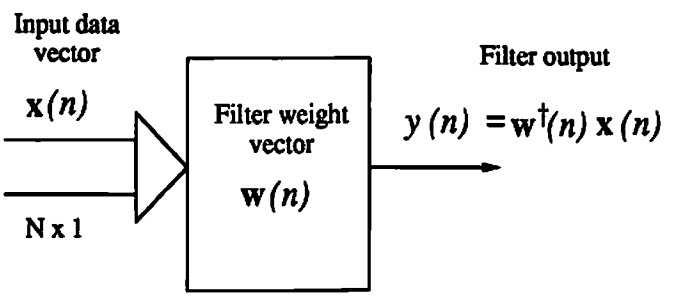

Figure 2.2: The block diagram of the direct form multi-channel FIR filter.

vector in this way is convenient since it expresses the output as the inner product of two vectors in both complex and real cases.

This multi-channel filter is represented by the block diagram in Figure 2.2. The input data, $\mathbf{x}(n)$, is a N-dimensional ($N = KL$) vector and the filter output, $y(n)$, is the inner product of $\mathbf{x}(n)$ and the weight vector $\mathbf{w}(n)$. Depending upon the particular application, the optimal weight vector $\mathbf{w}(n)$ is selected according to some specified criterion.

2.2.2 Linearly-Constrained Filter

Linearly-constrained minimum variance is a commonly used criterion: the power of the filter output is minimized subject to a number of linear constraints on the weight vector. The resulting filter is termed the linearly-constrained filter. To formulate the linearly-constrained filter, we define a minimization problem as

$$\min_{\mathbf{w}} \ \mathbf{w}^{\dagger}\mathbf{R}_{xx}\mathbf{w} \ , \tag{2.4}$$

subject to M linear constraints,

$$\mathbf{C}^{\dagger}\mathbf{w} = \mathbf{f} \ . \tag{2.5}$$

The term $\mathbf{R}_{xx}$ is the covariance matrix of the N-dimensional complex random input vector $\mathbf{x}(n)$ (assumed zero mean). The constraint matrix $\mathbf{C}$ contains M possibly complex column vectors while $\mathbf{f}$ specifies the corresponding constraint value for each vector. Without loss of generality, the matrix $\mathbf{C}$ is assumed full rank.

Although the matrix $\mathbf{C}$ may not necessary be an orthogonal matrix in general, it is sometimes more convenient to express $\mathbf{C}$ as an orthogonal matrix. (By orthogonal matrix we mean a matrix with orthonormal columns.) In the following, we outline the procedure for reformulating the constraint equations so that the constraint matrix is orthogonal.

Let the QR decomposition of $\mathbf{C}$ be

$$\mathbf{C} = \mathbf{QR} \tag{2.6}$$

where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is a triangular matrix. Substituting this equation into the constraint equation $\mathbf{C}^\dagger\mathbf{w} = \mathbf{f}$, we have

$$\mathbf{R}^\dagger\mathbf{Q}^\dagger\mathbf{w} = \mathbf{f}. \tag{2.7}$$

The full rank property of $\mathbf{C}$ ensures that $\mathbf{R}$ is invertible. Thus, the above equation can be rewritten as

$$\mathbf{Q}^\dagger\mathbf{w} = (\mathbf{R}^\dagger)^{-1}\mathbf{f}. \tag{2.8}$$

This constraint equation with $(\mathbf{R}^\dagger)^{-1}\mathbf{f}$ as the response vector and $\mathbf{Q}$ as the constraint matrix is equivalent to the original constraint equation. Thus, we can assume that the constraint matrix is orthogonal without loss of generality.

It was proved in [9] that the optimal solution to the minimization problem defined in Eq.(2.4) and (2.5) is

$$\mathbf{w}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{C}(\mathbf{C}^\dagger\mathbf{R}_{xx}^{-1}\mathbf{C})^{-1}\mathbf{f}. \tag{2.9}$$

Using this equation, we can compute the optimal weights if the terms $\mathbf{R}_{xx}$, $\mathbf{C}$ and $\mathbf{f}$ are given. In most practical applications, however, the signal environment in which the filter operates is unknown or time-varying and signal statistics such as signal covariance matrix cannot be determined in advance. It is therefore necessary to update the weights in the linearly-constrained filter according to the current signal environment. As a result, the design of the linearly-constrained filter essentially consists of two parts. The first is to set up the constraint equation specified by $\mathbf{C}$ and $\mathbf{f}$ using some *a priori* information about the signal environment; the second is to construct a rule to adjust the filter weights according to the current signal environment.

2.3 Specification of Constraint Equations

The purpose of the linearly-constrained filter is to reject the jammer and white noise components of $\mathbf{x}$ while retaining the desired signal portion. The significant feature of the linearly-constrained filter is that a portion of the filter response is held fixed and the remainder allowed to adapt. The dimension of the weight vector determines the

total degrees of freedom of the filter response. Some of the degrees of freedom are reduced by a set of linear constraints on the filter coefficients. The constraints are selected according to given prior knowledge and the remaining degrees of freedom are used to adjust the filter coefficients during operation in the actual signal environment. Specification of the constraints is based on a fixed-weight system design which uses the best available *a priori* information. Essentially, this process is identical to that conducted when system designers specify a non-adaptive filter structure. This section illustrates two examples of constraint specification for the cases of narrow-band and broad-band desired signals. More details on constraint specifications can be found in [10, 17, 18, 12, 13].

2.3.1 Narrow Band Signal

Consider the case in which the desired signal $\mathbf{s}$ is a single-source narrow band signal. Its covariance matrix is a rank one matrix represented by

$$\mathbf{R}_{ss} = \sigma_s^2 \mathbf{v}\mathbf{v}^\dagger. \tag{2.10}$$

The filter output power contributed by $\mathbf{s}$ then equals

$$\mathbf{w}^\dagger \mathbf{R}_{ss} \mathbf{w} = \sigma_s^2 |\mathbf{v}^\dagger \mathbf{w}|^2. \tag{2.11}$$

A typical constraint for this case requires

$$\mathbf{v}^\dagger \mathbf{w} = 1. \tag{2.12}$$

Observe that the output power contributed by $\mathbf{s}$ is held fixed under this constraint. Thus, minimizing the total output power subject to the constraint minimizes the noise power at the filter output.

Equation (2.12) requires the filter response to have unity gain at the direction specified by $\mathbf{v}$. Specification of $\mathbf{v}$ depends upon the particular application. For instance, in array processing the vector $\mathbf{v}$ corresponds to the direction that $\mathbf{s}$ impinges on the array. In this case, the knowledge of array geometry and the arrival direction of $\mathbf{s}$ are sufficient to compute $\mathbf{v}$.

2.3.2 Broad Band Signal

In the broad band case the covariance matrix $\mathbf{R}_{ss}$ is, in general, a positive definite matrix of rank N which can be represented by

$$\mathbf{R}_{ss} = \mathbf{V}\Sigma\mathbf{V}^{\dagger} \tag{2.13}$$

where the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{R}_{ss}$ and the diagonal elements in the diagonal matrix Σ are the corresponding eigenvalues of $\mathbf{R}_{ss}$. The concept of eigenvector constraint was introduced in [13] by choosing the M eigenvectors corresponding to the M most significant eigenvalues of $\mathbf{R}_{ss}$ to form the constraint matrix $\mathbf{C}$. The response vector $\mathbf{f}$ is then chosen as the best least-squares estimate of a pre-specified desired filter response. The *a priori* information needed to set up this type of constraint requires knowledge of $\mathbf{R}_{ss}$. An estimate of $\mathbf{R}_{ss}$ can be used if the exact $\mathbf{R}_{ss}$ is not available.

2.4 Fundamental Subspaces and Closed Form Solutions

In general, the input data vector $\mathbf{x}$ in the linearly-constrained filter can be expressed as the sum of three components,

$$\mathbf{x} = \mathbf{s} + \mathbf{j} + \mathbf{n} \tag{2.14}$$

consisting of the desired signal ($\mathbf{s}$), the jammer ($\mathbf{j}$) and the white noise ($\mathbf{n}$). These three components are assumed to be zero-mean and statistically independent. The covariance matrix of $\mathbf{x}$ therefore equals the sum of the individual covariance matrices, i.e.,

$$\mathbf{R}_{xx} = \mathbf{R}_{ss} + \mathbf{R}_{jj} + \sigma_n^2\mathbf{I} \tag{2.15}$$

where σ_n^2 is the white noise power.

In this dissertation, the investigation of the linearly-constrained filter is based upon the analysis of vector spaces induced from the covariance matrix $\mathbf{R}_{xx}$ and the

constraint matrix C. To facilitate the presentation thereafter, four fundamental subspaces and several closed form expressions for the optimal weights in the linearly-constrained filter are examined in this section.

2.4.1 Four Fundamental Subspaces

The four fundamental subspaces in the linearly-constrained filter are defined as follows:

Signal space: The vector space spanned by the columns of $\mathbf{R}_{ss}$.

Jammer space: The vector space spanned by the columns of $\mathbf{R}_{jj}$.

Constraint space: The vector space spanned by the columns of C.

Constraint Orthogonal space: The orthogonal complement subspace of the constraint space.

A full rank matrix $\mathbf{W}_s$ satisfying the condition

$$\mathbf{W}_s^\dagger \mathbf{C} = \mathbf{0} \tag{2.16}$$

is used to specify the constraint orthogonal space and is termed the *blocking matrix.*

2.4.2 Closed Form Solutions

In addition to the closed-form solution in Eq.(2.9), $\mathbf{w}_{opt}$ can be expressed in the alternative form

$$\mathbf{w}_{opt} = \mathbf{w}_q - \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{w}_q \tag{2.17}$$

where

$$\mathbf{w}_q = \mathbf{C}(\mathbf{C}^\dagger \mathbf{C})^{-1} \mathbf{f} \tag{2.18}$$

is the optimal solution under conditions of white noise input and $\mathbf{W}_s$ is the blocking matrix. Unlike Eq.(2.9), Eq.(2.17) expresses the optimal weights using the blocking matrix $\mathbf{W}_s$ instead of the constraint matrix C. In the next chapter, this expression

plays an important role in developing an effective iterative algorithm, the Generalized Sidelobe Canceller (GSC), to implement the adaptive linearly-constrained filter.

The alternative optimal weight expression is based on the following matrix identity:

$$\mathbf{I} - \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{R}_{xx} = \mathbf{R}_{xx}^{-1}\mathbf{C}(\mathbf{C}^\dagger \mathbf{R}_{xx}^{-1}\mathbf{C})^{-1}\mathbf{C}^\dagger. \tag{2.19}$$

The proof of this identity is given as follows. Let $\mathbf{H}$ be the square root of $\mathbf{R}_{xx}$, i.e.,

$$\mathbf{R}_{xx} = \mathbf{H}\mathbf{H}^\dagger. \tag{2.20}$$

The left-hand side of the matrix identity can be rewritten as

$$\begin{aligned}
& \mathbf{I} - \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{R}_{xx} && (2.21)\\
= \; & (\mathbf{H}^\dagger)^{-1}\mathbf{H}^\dagger - \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{H}\mathbf{H}^\dagger \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{H}\mathbf{H}^\dagger && (2.22)\\
= \; & (\mathbf{H}^\dagger)^{-1}[\mathbf{I} - \mathbf{H}^\dagger \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{H}\mathbf{H}^\dagger \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{H}]\mathbf{H}^\dagger && (2.23)\\
= \; & (\mathbf{H}^\dagger)^{-1}[\mathbf{I} - \mathbf{P}_{H^\dagger W_s}]\mathbf{H}^\dagger && (2.24)
\end{aligned}$$

where $\mathbf{P}_{H^\dagger W_s}$ is the projection matrix specified by $\mathbf{H}^\dagger \mathbf{W}_s$. And the right-hand side of the matrix identity can be rewritten as

$$\begin{aligned}
& \mathbf{R}_{xx}^{-1}\mathbf{C}(\mathbf{C}^\dagger \mathbf{R}_{xx}^{-1}\mathbf{C})^{-1}\mathbf{C}^\dagger && (2.25)\\
= \; & (\mathbf{H}\mathbf{H}^\dagger)^{-1}\mathbf{C}[\mathbf{C}^\dagger(\mathbf{H}\mathbf{H}^\dagger)^{-1}\mathbf{C}]^{-1}\mathbf{C}^\dagger && (2.26)\\
= \; & (\mathbf{H}^\dagger)^{-1}\mathbf{H}^{-1}\mathbf{C}[\mathbf{C}^\dagger(\mathbf{H}^{-1})^\dagger \mathbf{H}^{-1}\mathbf{C}]^{-1}\mathbf{C}^\dagger(\mathbf{H}^{-1})^\dagger \mathbf{H}^\dagger && (2.27)\\
= \; & (\mathbf{H}^\dagger)^{-1}\mathbf{P}_{H^{-1}C}\mathbf{H}^\dagger && (2.28)
\end{aligned}$$

where $\mathbf{P}_{H^{-1}C}$ is the projection matrix specified by $\mathbf{H}^{-1}\mathbf{C}$. Since

$$\begin{aligned}
(\mathbf{H}^\dagger \mathbf{W}_s)^\dagger \mathbf{H}^{-1}\mathbf{C} &= \mathbf{W}_s^\dagger \mathbf{C} && (2.29)\\
&= \mathbf{0}, && (2.30)
\end{aligned}$$

It follows that

$$\mathbf{I} - \mathbf{P}_{H^\dagger W_s} = \mathbf{P}_{H^{-1}C}. \tag{2.31}$$

Equations (2.24) and (2.28) are then equal and the proof of the matrix identity is completed. Multiplying both sides of Eq.(2.19) by $\mathbf{w}_q$ proves that Eq.(2.9) and (2.17) are equal.

From the proof of the matrix identity, the optimal weights of the linearly-constrained filter can be expressed as

$$\mathbf{w}_{opt} = (\mathbf{H}^\dagger)^{-1}\mathbf{P}_{H^{-1}C}\mathbf{H}^\dagger\mathbf{w}_q \quad (2.32)$$

$$= (\mathbf{H}^\dagger)^{-1}[\mathbf{I} - \mathbf{P}_{H^\dagger W_o}]\mathbf{H}^\dagger\mathbf{w}_q. \quad (2.33)$$

These two forms will be used in Chapter 6 to derive the generalized linearly-constrained filter.

2.5 Signal-to-Noise Ratio

A particularly important performance measure for a linearly-constrained filter is the output signal-to-noise ratio (SNR) when the filter has converged to the optimal weights. As shown in Eq.(2.17), the optimal weight vector can be split into the sum of two components. The first component, $\mathbf{w}_q$, is independent of $\mathbf{R}_{xx}$ and therefore exists as an explicit fixed-weight entity in the filter. Consequently, when $\mathbf{R}_{xx}$ is unknown only the second component needs to be implemented adaptively. (Iterative algorithms will be discussed in the next chapter.) The improvement of the SNR yielded by adaptively searching for the optimal weights according to the input data statistics can therefore be examined by comparing the SNR of the filter with $\mathbf{w}_{opt}$ and $\mathbf{w}_q$ as the filter weights. The understanding of this improvement is important since the system designers should consider this improvement as the cost-benefit trade-off factor when comparing the use of fixed-weight conventional filter versus adaptive linearly-constrained filter.

In Section 2.5.1, we derive the closed-form expressions for the SNR of the filter with $\mathbf{w}_{opt}$ and $\mathbf{w}_q$ as weights, respectively. A detailed discussion of the lower and upper bounds of the optimal SNR for all possible signal environment is given in Section 2.5.2.

2.5.1 Closed-Form Expressions

Definitions and Assumptions

The optimal weight vector can be decomposed into two components:

$$\mathbf{w}_{opt} = \mathbf{w}_q - \mathbf{w}_o \tag{2.34}$$

where $\mathbf{w}_q$ is the quiescent weight vector and $\mathbf{w}_o$ is the negative projection of $\mathbf{w}_{opt}$ onto the constraint orthogonal space, i.e.,

$$\mathbf{w}_o = -\mathbf{P}_{ws}\mathbf{w}_{opt}. \tag{2.35}$$

The term $\mathbf{P}_{ws}$ denotes the projection matrix specified by the blocking matrix $\mathbf{W}_s$. From Eq.(2.17) it follows that

$$\mathbf{w}_o = \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{R}_{xx}\mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger \mathbf{R}_{xx}\mathbf{w}_q. \tag{2.36}$$

Note that decomposing $\mathbf{w}_{opt}$ in this way separates the optimal weight vector into a data independent portion $\mathbf{w}_q$ and a data dependent portion $\mathbf{w}_o$.

Let the covariance matrix $\mathbf{R}_{xx}$ be the sum of two components corresponding to its signal and noise portions:

$$\mathbf{R}_{xx} = \mathbf{R}_{ss} + \mathbf{R}_{nn} \tag{2.37}$$

where the term

$$\mathbf{R}_{nn} = \mathbf{R}_{jj} + \sigma_n^2\mathbf{I} \tag{2.38}$$

corresponds to jammers and white noise. Denote the ranks of $\mathbf{R}_{ss}$ and $\mathbf{R}_{jj}$ as

$$r_s = \text{rank}(\mathbf{R}_{ss}) \quad \text{and} \quad r_j = \text{rank}(\mathbf{R}_{jj}). \tag{2.39}$$

Assuming the matrices $\mathbf{R}_{ss}$ and $\mathbf{R}_{jj}$ are non-negative definite, we can therefore decompose them as

$$\mathbf{R}_{ss} = \mathbf{H}_s\mathbf{H}_s^\dagger \quad \text{and} \quad \mathbf{R}_{jj} = \mathbf{H}_j\mathbf{H}_j^\dagger, \tag{2.40}$$

where $\mathbf{H}_s$ and $\mathbf{H}_j$ are both full rank and of sizes $N \times r_s$ and $N \times r_j$, respectively. To avoid $\mathbf{R}_{xx}$ to be singular, we also assume that σ_n^2 is non-zero.

Define the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ as

$$\mathbf{A} = \mathbf{P}_{ws}\mathbf{H}_j \tag{2.41}$$

and

$$\mathbf{b} = \mathbf{H}_j^{\dagger}\mathbf{w}_q. \tag{2.42}$$

The rank of $\mathbf{A}$, denoted by r, must satisfy

$$r \leq \min(r_j, N - M), \tag{2.43}$$

where $N - M$ is the rank of $\mathbf{P}_{ws}$. Let the singular value decomposition of $\mathbf{A}$ be

$$\mathbf{A} = \mathbf{U}_a\mathbf{\Sigma}_a\mathbf{V}_a^{\dagger} \tag{2.44}$$

where $\mathbf{\Sigma}_a$ is a $N \times r_j$ matrix of the form

$$\mathbf{\Sigma}_a = \begin{bmatrix} \mathbf{\Sigma}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.45}$$

The matrix $\mathbf{\Sigma}_r$ is a diagonal matrix containing the non-zero singular values of $\mathbf{A}$ and is denoted by

$$\mathbf{\Sigma}_r = diag(\sigma_1, \sigma_2, \cdots, \sigma_r). \tag{2.46}$$

To simplify the derivation of the signal to noise ratio, two assumptions are made. The first assumption is that the projection of the signal $\mathbf{s}$ onto the constraint orthogonal space is zero, that is,

$$\mathbf{P}_{ws}\mathbf{H}_s = \mathbf{0}. \tag{2.47}$$

This assumption is satisfied if all the power of $\mathbf{s}$ is preserved in the constraint space. If the rank of $\mathbf{H}_s$ is less than or equal to the number of constraints (M), it is possible to select the constraint matrix $\mathbf{C}$ so that this assumption is fully satisfied. Otherwise, we can choose $\mathbf{C}$ so that the assumption is approximately satisfied when $\mathbf{R}_{ss}$ has no more than M *significant* eigenvalues. The second assumption is that the blocking matrix is orthogonal, i.e.,

$$\mathbf{W}_s^{\dagger}\mathbf{W}_s = \mathbf{I}. \tag{2.48}$$

We can make this assumption without loss of generality since the SNR depends on $\mathbf{w}_{opt}$ but not the specific form of $\mathbf{W}_s$.

Expression for $\mathbf{w}_o$

Theorem 1 *Under the two assumptions mentioned above, $\mathbf{w}_o$ can be expressed as*

$$\mathbf{w}_o = \mathbf{U}_a \mathbf{S} \mathbf{V}_a^\dagger \mathbf{b} \tag{2.49}$$

where $\mathbf{S}$ is a $N \times r_j$ matrix of the form

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.50}$$

And the diagonal matrix $\mathbf{S}_r$ has positive diagonal elements given by,

$$\mathbf{S}_r = diag(\frac{\sigma_1}{\sigma_1^2+\sigma_n^2}, \frac{\sigma_2}{\sigma_2^2+\sigma_n^2}, \cdots, \frac{\sigma_r}{\sigma_r^2+\sigma_n^2}). \tag{2.51}$$

Proof: Using the two assumptions mentioned above and the fact that $\mathbf{W}_s^\dagger \mathbf{w}_q = \mathbf{0}$, it is easy to verify that

$$\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s = \sigma_n^2 \mathbf{I} + \mathbf{W}_s^\dagger \mathbf{R}_{jj} \mathbf{W}_s \tag{2.52}$$

and

$$\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{w}_q = \mathbf{W}_s^\dagger \mathbf{R}_{jj} \mathbf{w}_q. \tag{2.53}$$

Thus, the expression for $\mathbf{w}_o$ in Eq.(2.36) becomes

$$\mathbf{w}_o = \mathbf{W}_s (\sigma_n^2 \mathbf{I} + \mathbf{W}_s^\dagger \mathbf{R}_{jj} \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{R}_{jj} \mathbf{w}_q \tag{2.54}$$

or, equivalently,

$$\mathbf{w}_o = \mathbf{W}_s (\sigma_n^2 \mathbf{I} + \mathbf{W}_s^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{w}_q. \tag{2.55}$$

Using the Sherman-Morrison formula [19], the matrix inverse in the above equation equals

$$\frac{1}{\sigma_n^2}\mathbf{I} - \frac{1}{\sigma_n^2} \mathbf{W}_s^\dagger \mathbf{H}_j (\sigma_n^2 \mathbf{I} + \mathbf{H}_j^\dagger \mathbf{W}_s \mathbf{W}_s^\dagger \mathbf{H}_j)^{-1} \mathbf{H}_j^\dagger \mathbf{W}_s. \tag{2.56}$$

Substituting this into Eq.(2.55) shows that

$$\mathbf{w}_o = \frac{1}{\sigma_n^2} \mathbf{W}_s \mathbf{W}_s^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{w}_q - \frac{1}{\sigma_n^2} \mathbf{W}_s \mathbf{W}_s^\dagger \mathbf{H}_j (\sigma_n^2 \mathbf{I} + \mathbf{H}_j^\dagger \mathbf{W}_s \mathbf{W}_s^\dagger \mathbf{H}_j)^{-1} \mathbf{H}_j^\dagger \mathbf{W}_s \mathbf{W}_s^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{w}_q. \tag{2.57}$$

From the definitions of $\mathbf{A}$ and $\mathbf{b}$ and the fact that

$$\mathbf{P}_{ws} = \mathbf{W}_s \mathbf{W}_s^\dagger, \tag{2.58}$$

Equation (2.57) becomes

$$\mathbf{w}_o = \frac{1}{\sigma_n^2}\mathbf{A}[\mathbf{I} - (\sigma_n^2\mathbf{I} + \mathbf{A}^\dagger\mathbf{A})^{-1}\mathbf{A}^\dagger\mathbf{A}]\mathbf{b}\ . \tag{2.59}$$

Using the singular value decomposition of $\mathbf{A}$, the bracketed term of the above equation simplifies to

$$\mathbf{V}_a \mathbf{D} \mathbf{V}_a^\dagger, \tag{2.60}$$

where $\mathbf{D}$ is a $r_j \times r_j$ diagonal matrix given by

$$\mathbf{D} = diag(\frac{\sigma_n^2}{\sigma_1^2+\sigma_n^2}, \frac{\sigma_n^2}{\sigma_2^2+\sigma_n^2}, \cdots, \frac{\sigma_n^2}{\sigma_r^2+\sigma_n^2}, 1, \cdots, 1). \tag{2.61}$$

Therefore,

$$\mathbf{w}_o = \frac{1}{\sigma_n^2}\mathbf{A}\mathbf{V}_a\mathbf{D}\mathbf{V}_a^\dagger\mathbf{b}. \tag{2.62}$$

In addition, since

$$\mathbf{A}\mathbf{V}_a = \mathbf{U}_a\mathbf{\Sigma}_a, \tag{2.63}$$

the above equation becomes

$$\mathbf{w}_o = \frac{1}{\sigma_n^2}\mathbf{U}_a\mathbf{\Sigma}_a\mathbf{D}\mathbf{V}_a^\dagger\mathbf{b}. \tag{2.64}$$

Since the diagonal matrix $\mathbf{S}$ defined in Eq.(2.50) satisfies

$$\mathbf{S} = \frac{1}{\sigma_n^2}\mathbf{\Sigma}_a\mathbf{D}, \tag{2.65}$$

the vector $\mathbf{w}_o$ can be expressed in the form of Eq.(2.49).

Q.E.D.

Expressions for SNR

Theorem 2 *Let the SNR of the linearly-constrained filter with* $\mathbf{w}$ *as the weight vector be defined as*

$$SNR = \frac{\mathbf{w}^\dagger\mathbf{R}_{ss}\mathbf{w}}{\mathbf{w}^\dagger\mathbf{R}_{nn}\mathbf{w}}. \tag{2.66}$$

Then, the closed-form expressions for the SNR with $\mathbf{w} = \mathbf{w}_{opt}$ $\mathbf{w} = \mathbf{w}_q$ *are given by*

$$(a) \quad SNR_{opt} = \frac{\mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_q}{\mathbf{w}_q^\dagger (\sigma_n^2 \mathbf{I} + \mathbf{H}_j \mathbf{V}_a \mathbf{D} \mathbf{V}_a^\dagger \mathbf{H}_j^\dagger) \mathbf{w}_q}, \tag{2.67}$$

$$(b) \quad SNR_q = \frac{\mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_q}{\mathbf{w}_q^\dagger (\sigma_n^2 \mathbf{I} + \mathbf{H}_j \mathbf{H}_j^\dagger) \mathbf{w}_q}. \tag{2.68}$$

Proof: Using the decomposition in Eq.(2.34) for $\mathbf{w}_{opt}$, we can express the output power contributed by the signal as

$$\mathbf{w}_{opt}^\dagger \mathbf{R}_{ss} \mathbf{w}_{opt} = \mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_q + \mathbf{w}_o^\dagger \mathbf{R}_{ss} \mathbf{w}_o - \mathbf{w}_o^\dagger \mathbf{R}_{ss} \mathbf{w}_q - \mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_o. \tag{2.69}$$

The last three terms are zero because of the assumption that $\mathbf{P}_{ws} \mathbf{H}_s = \mathbf{0}$. Thus,

$$\mathbf{w}_{opt}^\dagger \mathbf{R}_{ss} \mathbf{w}_{opt} = \mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_q. \tag{2.70}$$

We can also express the output power contributed by the noise as

$$\mathbf{w}_{opt}^\dagger \mathbf{R}_{nn} \mathbf{w}_{opt} = \sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q + \sigma_n^2 \mathbf{w}_o^\dagger \mathbf{w}_o + \mathbf{w}_{opt}^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{w}_{opt}. \tag{2.71}$$

Since

$$\begin{aligned} \mathbf{H}_j^\dagger \mathbf{w}_{opt} &= \mathbf{H}_j^\dagger \mathbf{w}_q - \mathbf{H}_j^\dagger \mathbf{w}_o && (2.72) \\ &= \mathbf{b} - \mathbf{H}_j^\dagger \mathbf{w}_o && (2.73) \\ &= \mathbf{b} - \mathbf{H}_j^\dagger \mathbf{P}_{ws} \mathbf{w}_o && (2.74) \\ &= \mathbf{b} - \mathbf{A}^\dagger \mathbf{w}_o, && (2.75) \end{aligned}$$

the last term of Eq.(2.71) equals

$$||\mathbf{b} - \mathbf{A}^\dagger \mathbf{w}_o||_2^2 \tag{2.76}$$

where $||\cdot||_2$ denotes the vector 2-norm. As a result, the signal to noise ratio when $\mathbf{w} = \mathbf{w}_{opt}$ is given by

$$SNR_{opt} = \frac{\mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_q}{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q + \sigma_n^2 \mathbf{w}_o^\dagger \mathbf{w}_o + ||\mathbf{b} - \mathbf{A}^\dagger \mathbf{w}_o||_2^2}. \tag{2.77}$$

The above SNR expression can be further simplified by substituting $\mathbf{w}_o$ in Eq.(2.49) into the last two terms of the denominator. The first of these terms becomes

$$\begin{aligned} \sigma_n^2 \mathbf{w}_o^\dagger \mathbf{w}_o &= \sigma_n^2 \mathbf{b}^\dagger \mathbf{V}_a \mathbf{S}^\dagger \mathbf{U}_a^\dagger \mathbf{U}_a \mathbf{S} \mathbf{V}_a^\dagger \mathbf{b} && (2.78) \\ &= \sigma_n^2 \mathbf{b}^\dagger \mathbf{V}_a \mathbf{S}^2 \mathbf{V}_a^\dagger \mathbf{b} && (2.79) \end{aligned}$$

since $\mathbf{U}_a$ is an orthogonal matrix. (In the above expression, $\mathbf{S}^2$ is an abbreviated notation for $\mathbf{S}^\dagger\mathbf{S}$. This abbreviation will be used hereafter without further notice.) In addition, since

$$\begin{aligned} \mathbf{b} - \mathbf{A}^\dagger \mathbf{w}_o &= \mathbf{b} - \mathbf{A}^\dagger \mathbf{U}_a \mathbf{S} \mathbf{V}_a^\dagger \mathbf{b} && (2.80) \\ &= \mathbf{b} - \mathbf{V}_a \mathbf{\Sigma}_a^\dagger \mathbf{U}_a^\dagger \mathbf{U}_a \mathbf{S} \mathbf{V}_a^\dagger \mathbf{b} && (2.81) \\ &= \mathbf{b} - \mathbf{V}_a \mathbf{\Sigma}_a^\dagger \mathbf{S} \mathbf{V}_a^\dagger \mathbf{b} && (2.82) \\ &= \mathbf{V}_a (\mathbf{I} - \mathbf{\Sigma}_a^\dagger \mathbf{S}) \mathbf{V}_a^\dagger \mathbf{b}, && (2.83) \end{aligned}$$

the second term becomes

$$||\mathbf{b} - \mathbf{A}^\dagger \mathbf{w}_o||_2^2 = \mathbf{b}^\dagger \mathbf{V}_a (\mathbf{I} - \mathbf{\Sigma}_a^\dagger \mathbf{S})^2 \mathbf{V}_a^\dagger \mathbf{b}. \quad (2.84)$$

Therefore,

$$\begin{aligned} \sigma_n^2 \mathbf{w}_o^\dagger \mathbf{w}_o + ||\mathbf{b} - \mathbf{A}^\dagger \mathbf{w}_o||_2^2 &= \mathbf{b}^\dagger \mathbf{V}_a [\sigma_n^2 \mathbf{S}^2 + (\mathbf{I} - \mathbf{\Sigma}_a^\dagger \mathbf{S})^2] \mathbf{V}_a^\dagger \mathbf{b} && (2.85) \\ &= \mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{V}_a [\sigma_n^2 \mathbf{S}^2 + (\mathbf{I} - \mathbf{\Sigma}_a^\dagger \mathbf{S})^2] \mathbf{V}_a^\dagger \mathbf{H}_j^\dagger \mathbf{w}_q && (2.86) \end{aligned}$$

The bracketed matrix sum in above equation is a diagonal matrix which can be verified to be equal to the matrix $\mathbf{D}$ defined in Eq.(2.61). Thus, the SNR_{opt} can be expressed as in Eq.(2.67).

From Eqs.(2.38) and (2.40),

$$\mathbf{w}_q^\dagger \mathbf{R}_{nn} \mathbf{w}_q = \mathbf{w}_q^\dagger (\sigma_n^2 \mathbf{I} + \mathbf{H}_j \mathbf{H}_j^\dagger) \mathbf{w}_q. \quad (2.87)$$

It follows that the SNR for $\mathbf{w} = \mathbf{w}_q$ is given by Eq.(2.68).

Q.E.D.

2.5.2 Discussion

Equation (2.34) shows that the optimal weight vector of a linearly-constrained filter can always be decomposed into two components, $\mathbf{w}_q$ and $\mathbf{w}_o$. Since $\mathbf{w}_q$ is the optimal weight when the input vector $\mathbf{x}$ has no jammer, component $\mathbf{w}_o$ serves to reject the jammer but not the white noise. Hence, when the input vector consists of both jammer and white noise, using $\mathbf{w}_o$ increases white noise power at the filter output. $\mathbf{w}_o$, which provides minimum noise power output, finds the optimum trade-off between increased white noise power output and decreased jammer noise output. Clearly, this trade-off depends on the input jammer to white noise power ratio (JNR). In the following discussion, we study how this trade-off affects the output signal to noise ratio of the filter. To provide a quantitative study, we first consider the upper and lower limits on the output SNR for all possible noise environments. We then define two factors which measure the ratios of the actual SNR to its lower and upper limits, respectively. We show that these two factors can be bounded in terms of the JNR at the input and several other places in the filter. By estimating the JNR we can therefore use the bounds on the two factors to predict the performance of the adaptive linearly-constrained filter in a particular noise environment.

Jammer to white noise ratios

Consider the input noise covariance matrix $\mathbf{R}_{nn}$ in Eq.(2.38). We define the JNR at the filter input as the ratio of the maximum eigenvalue of $\mathbf{R}_{jj}$ to the white noise power, that is,

$$JNR_{in} = \frac{\lambda_{max}(\mathbf{R}_{jj})}{\sigma_n^2}. \tag{2.88}$$

We also define two other types of JNR in the filter, JNR_q and JNR_i, which are induced from the weight vector $\mathbf{w}_q$ and the projection matrix $\mathbf{P}_{ws}$, respectively. JNR_q is defined as

$$JNR_q = \frac{\mathbf{w}_q^\dagger \mathbf{R}_{jj} \mathbf{w}_q}{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q}. \tag{2.89}$$

and the JNR_i are defined as

$$JNR_i = \frac{\sigma_i^2}{\sigma_n^2} \quad \text{for } i = 1, 2, \cdots, r \tag{2.90}$$

where σ_i^2 is the i_th largest eigenvalue of the matrix $\mathbf{P}_{ws}\mathbf{R}_{jj}\mathbf{P}_{ws}$ (or $\mathbf{A}\mathbf{A}^\dagger$). The maximum value of JNR_i for $i = 1, 2, \cdots, r$ is denoted by JNR_{max}.

The lower and upper bounds of SNR_{opt}

By comparing Eq.(2.67) with Eq.(2.68), one can see that the only difference between SNR_{opt} and SNR_q is the second term in the denominator. Rewriting the second term in the denominator of SNR_q as

$$\mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{V}_a \mathbf{I} \mathbf{V}_a^\dagger \mathbf{H}_j^\dagger \mathbf{w}_q \tag{2.91}$$

shows that the corresponding term in SNR_{opt} can be obtained by replacing the identity matrix $\mathbf{I}$ with the diagonal matrix $\mathbf{D}$. The difference in SNR between using $\mathbf{w}_q$ and $\mathbf{w}_{opt}$ as the weight vectors of the linearly-constrained filter therefore depends on the diagonal matrix $\mathbf{D}$. Since the main diagonal elements of $\mathbf{D}$ are non-negative and have values less than or equal to 1,

$$SNR_{opt} \geq SNR_q \tag{2.92}$$

for any $\mathbf{R}_{xx}$.

On the other hand, we can bound SNR_{opt} by replacing the second term of the denominator of Eq.(2.67) with zero. This can occur when there is no jammer at the input or when the weight vector $\mathbf{w}_q$ is orthogonal to $\mathbf{H}_j$. As a result,

$$SNR_{opt} \leq SNR_{max} = \frac{\mathbf{w}_q^\dagger \mathbf{R}_{ss} \mathbf{w}_q}{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q}. \tag{2.93}$$

We define a factor δ_l as the ratio of SNR_{opt} to SNR_q:

$$\delta_l = \frac{SNR_{opt}}{SNR_q}. \tag{2.94}$$

This factor indicates the degree of SNR improvement due to the inclusion of $\mathbf{w}_o$ in the optimal weight. Similarly, we can define another factor δ_m as the ratio of SNR_{max} to SNR_{opt}:

$$\delta_m = \frac{SNR_{max}}{SNR_{opt}}. \tag{2.95}$$

which measures how far away is the SNR_{opt} from its upper bound. In the following, we derive the bounds for these two factors in terms of the jammer to white noise ratios as defined above.

Using Eq.(2.67) and Eq.(2.68), one has

$$\delta_l = \frac{\mathbf{w}_q^\dagger(\sigma_n^2\mathbf{I} + \mathbf{H}_j\mathbf{H}_j^\dagger)\mathbf{w}_q}{\mathbf{w}_q^\dagger(\sigma_n^2\mathbf{I} + \mathbf{H}_j\mathbf{V}_a\mathbf{D}\mathbf{V}_a^\dagger\mathbf{H}_j^\dagger)\mathbf{w}_q}. \tag{2.96}$$

In addition, using the fact that

$$\mathbf{D} = \mathbf{I} - \mathbf{\Sigma}_a^\dagger\mathbf{S} \tag{2.97}$$

the denominator of δ_l can be rewritten as

$$\mathbf{w}_q^\dagger(\sigma_n^2\mathbf{I} + \mathbf{H}_j\mathbf{H}_j^\dagger)\mathbf{w}_q - \mathbf{w}_q^\dagger\mathbf{H}_j\mathbf{V}_a\mathbf{\Sigma}_a^\dagger\mathbf{S}\mathbf{V}_a^\dagger\mathbf{H}_j^\dagger\mathbf{w}_q \tag{2.98}$$

where the numerator of δ_l appears explicitly as one term of the denominator. It is then easy to verify that δ_l can be expressed as

$$\delta_l = 1 + \frac{\alpha_l}{1-\alpha_l} \tag{2.99}$$

where

$$\alpha_l = \frac{\mathbf{w}_q^\dagger\mathbf{H}_j\mathbf{V}_a\mathbf{\Sigma}_a^\dagger\mathbf{S}\mathbf{V}_a^\dagger\mathbf{H}_j^\dagger\mathbf{w}_q}{\mathbf{w}_q^\dagger(\sigma_n^2\mathbf{I} + \mathbf{w}_q^\dagger\mathbf{H}_j\mathbf{H}_j^\dagger)\mathbf{w}_q}. \tag{2.100}$$

Alternatively, α_l can also be expressed as

$$\alpha_l = \frac{\mathbf{b}^\dagger\mathbf{V}_a\mathbf{\Sigma}_a^\dagger\mathbf{S}\mathbf{V}_a^\dagger\mathbf{b}/\mathbf{b}^\dagger\mathbf{b}}{\sigma_n^2\dfrac{\mathbf{w}_q^\dagger\mathbf{w}_q}{\mathbf{b}^\dagger\mathbf{b}} + 1} \tag{2.101}$$

where b is defined in Eq.(2.42). Since the non-zero terms of the diagonal matrix $\mathbf{\Sigma}_a^\dagger\mathbf{S}$ are given by

$$\frac{\sigma_i^2}{\sigma_i^2 + \sigma_n^2} = \frac{JNR_i}{1 + JNR_i} \quad \text{for } i = 1, 2, \cdots, r, \tag{2.102}$$

and the first term in the denominator of Eq.(2.101) is given by

$$\frac{1}{JNR_q}, \tag{2.103}$$

we have

$$\alpha_l \leq \frac{JNR_{max}}{1+JNR_{max}} \cdot \frac{JNR_q}{1+JNR_q}. \tag{2.104}$$

Since δ_l is a monotonically increasing function of α_l, it is correspondingly bounded by

$$\delta_l \leq 1 + \frac{JNR_{max} \cdot JNR_q}{1+JNR_{max}+JNR_q}. \tag{2.105}$$

In addition, since JNR_q and JNR_{max} are both no greater than JNR_{in}, δ_l can also be bounded by the input jammer to noise ratio as

$$\delta_l \leq 1 + \frac{JNR_{in}{}^2}{1+2JNR_{in}}. \tag{2.106}$$

Proceeding in a similar fashion, we can derive a bound on δ_m in terms of JNR_q and JNR_{in}. Directly substituting the expressions for SNR_{opt} and SNR_{max} in Eqs.(2.67) and (2.93), respectively, one has

$$\delta_m = \frac{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q + \mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{V}_a \mathbf{D} \mathbf{V}_a^\dagger \mathbf{H}_j^\dagger \mathbf{w}_q}{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q}. \tag{2.107}$$

Or, alternatively,

$$\delta_m = 1 + \frac{\mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{V}_a \mathbf{D} \mathbf{V}_a^\dagger \mathbf{H}_j^\dagger \mathbf{w}_q}{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q}. \tag{2.108}$$

To obtain a bound on δ_m, the second term of the above equation is expressed as a product of two terms:

$$\frac{\mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{V}_a \mathbf{D} \mathbf{V}_a^\dagger \mathbf{H}_j^\dagger \mathbf{w}_q}{\mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{w}_q} \cdot \frac{\mathbf{w}_q^\dagger \mathbf{H}_j \mathbf{H}_j^\dagger \mathbf{w}_q}{\sigma_n^2 \mathbf{w}_q^\dagger \mathbf{w}_q}. \tag{2.109}$$

The first of these terms is bounded by unity since the maximum eigenvalue of $\mathbf{V}_a \mathbf{D} \mathbf{V}_a^\dagger$ is less than or equal to one, and the second equals JNR_q by definition. Thus, the value of δ_m is bounded by,

$$\delta_m \leq 1 + JNR_q \tag{2.110}$$

$$\leq 1 + JNR_{in} \tag{2.111}$$

2.6 The Addition of Constraints

In this section, we discuss the effect of adding constraints to the linearly constrained filter. We derive recursive formulas to update the optimal weight and output power expression when one single additional constraint is added and provide an interpretation of these recursive formulas.

2.6.1 Recursive Formulas

Let the constraint equation of a linearly-constrained filter be denoted as

$$\mathbf{C}_i^\dagger \mathbf{w} = \mathbf{f}_i, \tag{2.112}$$

where $\mathbf{C}_i$ has i columns. We add one more constraint to this filter by appending a column vector $\mathbf{c}_{i+1}$ to $\mathbf{C}_i$ and an element f_{i+1} to $\mathbf{f}_i$. The new constraint equation is

$$\mathbf{C}_{i+1}^\dagger \mathbf{w} = \mathbf{f}_{i+1} \tag{2.113}$$

where

$$\mathbf{C}_{i+1} = [\mathbf{C}_i \quad \mathbf{c}_{i+1}] \tag{2.114}$$

and

$$\mathbf{f}_{i+1} = \begin{bmatrix} \mathbf{f}_i \\ f_{i+1} \end{bmatrix}. \tag{2.115}$$

The matrices $\mathbf{C}_i$ and $\mathbf{C}_{i+1}$ are assumed orthogonal for convenience.

Let $\mathbf{w}_i$ and $\mathbf{w}_{i+1}$ be the optimal weights of the filter before and after adding the constraint, respectively. It is shown in Appendix A that $\mathbf{w}_{i+1}$ can be obtained from $\mathbf{w}_i$ via the following recursive formula:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + (f_{i+1} - \mathbf{c}_{i+1}^\dagger \mathbf{w}_i)\mathbf{w}_{c_{i+1}} \tag{2.116}$$

where

$$\mathbf{w}_{c_{i+1}} = (\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger \mathbf{c}_{i+1}. \tag{2.117}$$

The term $\mathbf{P}_{i+1}$ is the projection matrix specified by $\mathbf{H}^{-1}\mathbf{C}_{i+1}$ and $\mathbf{H}$ is the square-root of the covariance matrix $\mathbf{R}_{xx}$. The vector $\mathbf{w}_{c_{i+1}}$ is actually the optimal weight of the

filter with the constraint,

$$\mathbf{C}_{i+1}^{\dagger}\mathbf{w} = \mathbf{e}_{i+1}, \tag{2.118}$$

where $\mathbf{e}_{i+1}$ is a vector having unity as its $(i+1)\_th$ element and zeros elsewhere.

Multiplying both sides of Eq.(2.116) by $\mathbf{H}^{\dagger}$, we have

$$\mathbf{H}^{\dagger}\mathbf{w}_{i+1} = \mathbf{H}^{\dagger}\mathbf{w}_i + (f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i)\mathbf{H}^{\dagger}\mathbf{w}_{c_{i+1}}. \tag{2.119}$$

The right-hand side of the above equation is the sum of two orthogonal vectors, which can be proved by examining their inner product,

$$\mathbf{w}_{c_{i+1}}^{\dagger}\mathbf{R}_{xx}\mathbf{w}_i. \tag{2.120}$$

The scaler term, $f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i$, is omitted in above expression since only the angle between the two vectors is important. Using the expression for the vector $\mathbf{w}_{c_{i+1}}$ in Eq.(2.117), one has

$$\begin{aligned} \mathbf{w}_{c_{i+1}}^{\dagger}\mathbf{R}_{xx}\mathbf{w}_i &= \mathbf{c}_{i+1}^{\dagger}\mathbf{H}\mathbf{P}_{i+1}\mathbf{H}^{-1}\mathbf{R}_{xx}\mathbf{w}_i & (2.121)\\ &= \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{w}_i. & (2.122) \end{aligned}$$

This can be further simplified using the fact that

$$(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{w}_i = \mathbf{w}_i, \tag{2.123}$$

which is also proven in Appendix A. Thus,

$$\begin{aligned} \mathbf{w}_{c_{i+1}}^{\dagger}\mathbf{R}_{xx}\mathbf{w}_i &= \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{w}_i & (2.124)\\ &= \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{R}_{xx}^{-1}\mathbf{C}_i(\mathbf{C}_i^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{C}_i)^{-1}\mathbf{f}_i & (2.125)\\ &= \mathbf{c}_{i+1}^{\dagger}\mathbf{C}_i(\mathbf{C}_i^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{C}_i)^{-1}\mathbf{f}_i & (2.126)\\ &= 0. & (2.127) \end{aligned}$$

Let the output power of the filter before and after adding the additional constraint be ε_i and ε_{i+1} respectively. A recursive formula for the output power can now be obtained by taking the L_2 norm on both sides of Eq.(2.119). Specifically,

$$\varepsilon_{i+1} = \varepsilon_i + |f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i|^2\varepsilon_{c_{i+1}} \tag{2.128}$$

where

$$\varepsilon_{c_{i+1}} = |\mathbf{H}^{\dagger}\mathbf{w}_{c_{i+1}}|^2. \tag{2.129}$$

2.6.2 Discussion

The update of the optimal weight

As revealed in Eq.(2.116), $\mathbf{w}_{i+1}$ is a linear combination of $\mathbf{w}_i$ and $\mathbf{w}_{c_{i+1}}$. The vector $\mathbf{w}_{c_{i+1}}$ and the scalar $f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i$ determine, respectively, at which direction and by what amount the optimal weight changes due to the addition of the constraint $\mathbf{c}_{i+1}^{\dagger}\mathbf{w} = f_{i+1}$. Since $\mathbf{w}_i$ lies in a space spanned by the basis $\{\mathbf{w}_{c1}, \mathbf{w}_{c2}, \cdots, \mathbf{w}_{ci}\}$, $\mathbf{w}_{i+1}$ lies correspondingly in a higher dimensional space spanned by a new basis which is formed by adding a linearly independent vector $\mathbf{w}_{c_{i+1}}$ to the previous basis. This additional basis vector, however, is not generally orthogonal to the space spanned by the previous basis. Figure 2.3 (a) shows conceptually the new optimal weight $\mathbf{w}_{i+1}$ as a linear combination of two unorthogonal vectors, $\mathbf{w}_i$ and $\mathbf{w}_{c_{i+1}}$. Using a transformation specified by $\mathbf{H}^{\dagger}$, however, the update of the optimal weight in the transformed space corresponds to adding one orthogonal vector to the previous optimal weight as shown in Figure 2.3 (b). This orthogonality update follows from the fact that the vector $\mathbf{H}^{\dagger}\mathbf{w}_i$ is always orthogonal to $\mathbf{H}^{\dagger}\mathbf{w}_{c_{i+1}}$ as shown in Eq.(2.127).

The change of output power

Figure 2.4 illustrates the increase in output power caused by the addition of constraints. The term $\varepsilon_k, k = 0, 1, \cdots, i+1$ denotes the output power for the filter with k constraints. The increase in the output power from the i_th to the $(i+1)\_th$ stage equals $\varepsilon_{c_{i+1}}$ multiplied by $|f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i|^2$.

The quiescent and optimal filter responses

It is interesting to observe the change in the quiescent and optimal filter responses due to the addition of a constraint. The quiescent response is defined as the optimal filter response under conditions of white noise input. It is easy to verify that the change in the quiescent filter weights due to the addition of one more constraint can be expressed as,

$$\mathbf{w}_{q_{i+1}} = \mathbf{w}_{qi} + f_{i+1}\mathbf{c}_{i+1}. \tag{2.130}$$

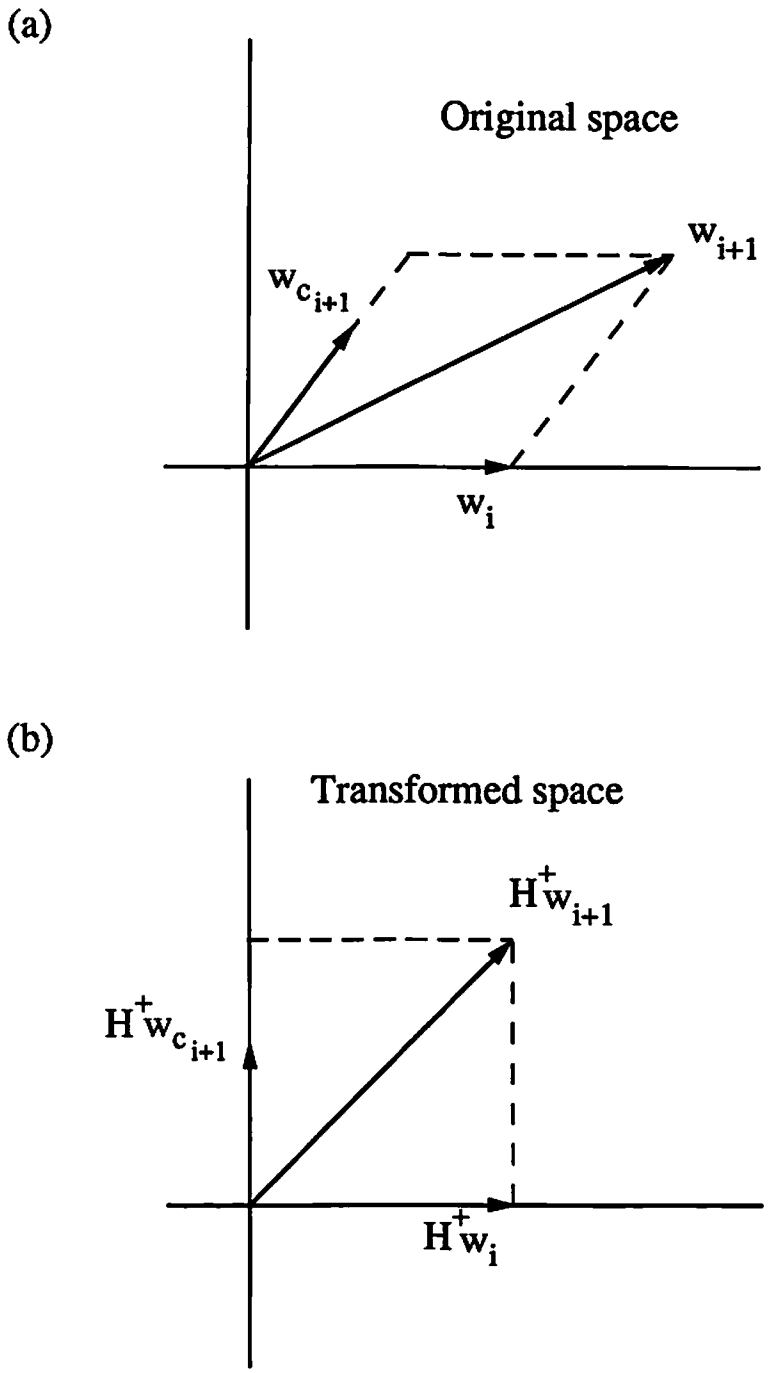

Figure 2.3: The conceptualized diagram of the weight vector update due to an additional constraint in (a) the original space, (b) the transformed space.

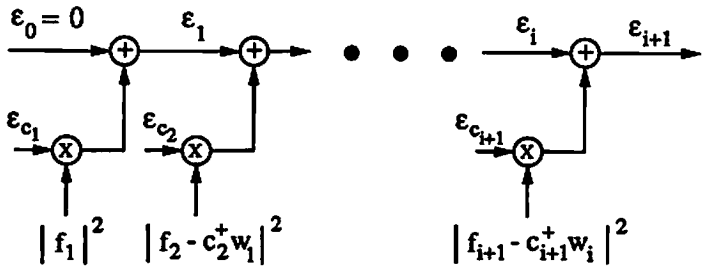

Figure 2.4: The increase in minimum output power for succeeding stages.

This is in fact a special case of Eq.(2.116) with $\mathbf{H} = \mathbf{I}$, the identity matrix.

When $f_{i+1} = 0$, the recursive formulas for the quiescent and optimal weights are

$$\mathbf{w}_{q_{i+1}} = \mathbf{w}_{qi} \tag{2.131}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i - (\mathbf{c}_{i+1}^\dagger \mathbf{w}_i)\mathbf{w}_{c_{i+1}}. \tag{2.132}$$

On the other hand, when $f_{i+1} = \mathbf{c}_{i+1}^\dagger \mathbf{w}_i$,

$$\mathbf{w}_{q_{i+1}} = \mathbf{w}_{qi} + (\mathbf{c}_{i+1}^\dagger \mathbf{w}_i)\mathbf{c}_{i+1} \tag{2.133}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i. \tag{2.134}$$

Thus, for an arbitrary vector $\mathbf{c}_{i+1}$ (orthogonal to $\mathbf{C}_i$), it is always possible to choose the corresponding f_{i+1} so that either the quiescent response or the optimal response remains unchanged due to the addition of the constraint.

2.7 Summary

This chapter has examined some important aspects of steady-state behavior of the linearly-constrained filter, including the optimal weights, signal-to-noise ratios, and the addition of constraints. While we have by no means completely studied the steady-state performance of linearly-constrained filters, we have made some progress in analyzing their behavior from a vector space standpoint. The results obtained in this chapter can be used to derive efficient adaptive implementation structures and to establish some guidelines for constraint specification.

The Generalized Sidelobe Canceller structure which adaptively implements the linearly-constrained filter in [11] in fact based on the closed-form expressions in Section 2.5. A detailed derivation of the GSC will be given in the next chapter and three iterative algorithms for implementing the linearly-constrained filter are compared. Further exploiting the closed-form expressions for the optimal weight results in a generalized linearly-constrained filter structure, termed the *Extended Generalized Sidelobe Canceller*, which will be presented in Chapter 6.

The SNR expressions in Section 2.5 are more general than those results obtained in [20] where only one constraint was allowed. In addition, by using a pre-assumed input covariance matrix, we can select constraints which maximize the resulting closed-form expression for the optimal SNR. Moreover, our discussion of the effect of adding linear constraints to the filter has application toward the design of partially adaptive arrays. Further investigations on these topics are required.

Chapter 3

Iterative Algorithms

3.1 Introduction

Several closed-form expressions for the optimal weights in the linearly-constrained filter have been presented in last chapter. Using any of these expressions, the optimal weights can be easily found provided that $\mathbf{R}_{xx}$, $\mathbf{C}$ and $\mathbf{f}$ are known. When applying the linearly-constrained filter to a practical signal processing problem, it is possible to set up the constraints specified by $\mathbf{C}$ and $\mathbf{f}$ using some priori knowledge about signal characteristics. However, in most practical situation the actual environment in which the filter is operated is unknown or varying in time. In such cases, the weights of the linearly-constrained filter cannot be pre-computed optimally and have to be adjusted using some adaptive algorithm. The resulting filter is termed an *adaptive linearly-constrained filter.*

The purpose of this chapter is to present and compare several iterative algorithms for realizing the linearly-constrained filter adaptively. Section 3.2 first derives three iterative algorithms: the *Constrained LMS (CLMS)*, the *Generalized Sidelobe Canceller (GSC)* and the *Modified Generalized Sidelobe Canceller (MGSC).* The first two algorithms were originally proposed in [9] and [11], respectively, in the context of array signal processing for implementing the antenna receivers; the last algorithm was derived in [21, 22] to simplify the implementation of the GSC algorithm. As opposed to original derivations, the approaches taken here to derive the CLMS and

GSC are based on a vector space decomposition which provides clear insights into these two algorithms; modifying the GSC algorithm by changing the coordinates of filter weights results in the MGSC algorithm. In Section 3.3, comparisons of the three algorithms from algebraic and geometrical points of view are presented. Two models, an implementation model and an analysis model, are established in Section 3.4 for depicting adaptive linearly-constrained filters. The former model, which decouples filter weights into fixed-weight and adaptive-weight portions, is useful for deriving effective implementation structures; and the latter model, which represents all classes of adaptive linearly-constrained filters, is useful for comparing the convergence performance due to different choices of the fixed-weight entities. In the final section, some concluding remarks are given.

3.2 Iterative Algorithms

For an adaptive linearly-constrained filter of size N with M constraints, the instantaneous weights form a vector, denoted by $\mathbf{w}(n)$, and the constraint matrix, denoted by $\mathbf{C}$, specifies a M-dimensional subspace in the N-dimensional vector space. To fully characterize the N-dimensional vector space, a blocking matrix, denoted by $\mathbf{W}_s$, whose columns span the orthogonal complement subspace of $\mathbf{C}$ is specified. The subspaces designated by $\mathbf{C}$ and $\mathbf{W}_s$ are termed as the constraint space and constraint orthogonal space, respectively. The following three algorithms are derived based on the decomposition of the instantaneous weights into these two orthogonal complement subspaces.

3.2.1 The Constrained LMS

The Algorithm

The weight vector $\mathbf{w}(n)$ can be uniquely decomposed as the sum of the projections in the two orthogonal complement subspaces:

$$\mathbf{w}(n) = \mathbf{P}_c\mathbf{w}(n) + \mathbf{P}_{ws}\mathbf{w}(n). \tag{3.1}$$

The terms $\mathbf{P}_c$ and $\mathbf{P}_{ws}$ denote the projection matrices specified by C and $\mathbf{W}_s$, respectively. Using the fact that

$$\mathbf{C}^\dagger \mathbf{w}(n) = \mathbf{f}, \tag{3.2}$$

it is easy to verify

$$\begin{aligned} \mathbf{P}_c \mathbf{w}(n) &= \mathbf{C}(\mathbf{C}^\dagger \mathbf{C})^{-1}\mathbf{f} && (3.3) \\ &= \mathbf{w}_q. && (3.4) \end{aligned}$$

Thus,

$$\mathbf{w}(n) = \mathbf{w}_q + \mathbf{P}_{ws}\mathbf{w}(n). \tag{3.5}$$

With the initial condition $\mathbf{w}(0) = \mathbf{w}_q$, the Constraint LMS [9] updates the weights in the N-dimensional space by,

$$\mathbf{w}(n+1) = \mathbf{w}_q + \mathbf{P}_{ws}[\mathbf{w}(n) + \mu y^*(n)\mathbf{x}(n)] \tag{3.6}$$

where $y(n) = \mathbf{w}^\dagger(n)\mathbf{x}(n)$ is the instantaneous system output and the superscript * is the notation for complex conjugate. This algorithm is *constrained* since the update term is restricted in the constraint orthogonal space by the matrix $\mathbf{P}_{ws}$. However, the algorithm can be rewritten in a unconstrained form as

$$\begin{aligned} \mathbf{w}_o(n+1) &= \mathbf{w}_o(n) + \mu y^*(n)\mathbf{x}_o(n) && (3.7) \\ \mathbf{w}(n+1) &= \mathbf{w}_q - \mathbf{w}_o(n+1) && (3.8) \end{aligned}$$

by defining

$$\mathbf{w}_o(n) = -\mathbf{P}_{ws}\mathbf{w}(n) \tag{3.9}$$

$$\mathbf{x}_o(n) = \mathbf{P}_{ws}\mathbf{x}(n). \tag{3.10}$$

In fact, this reformulation represents a more general version of the original CLMS since the initial weights need not start at $\mathbf{w}_o(0) = \mathbf{0}$. Unaware to the authors, the constraint elimination technique presented in [23] is actually the same as reformulating the CLMS in the above unconstrained form.

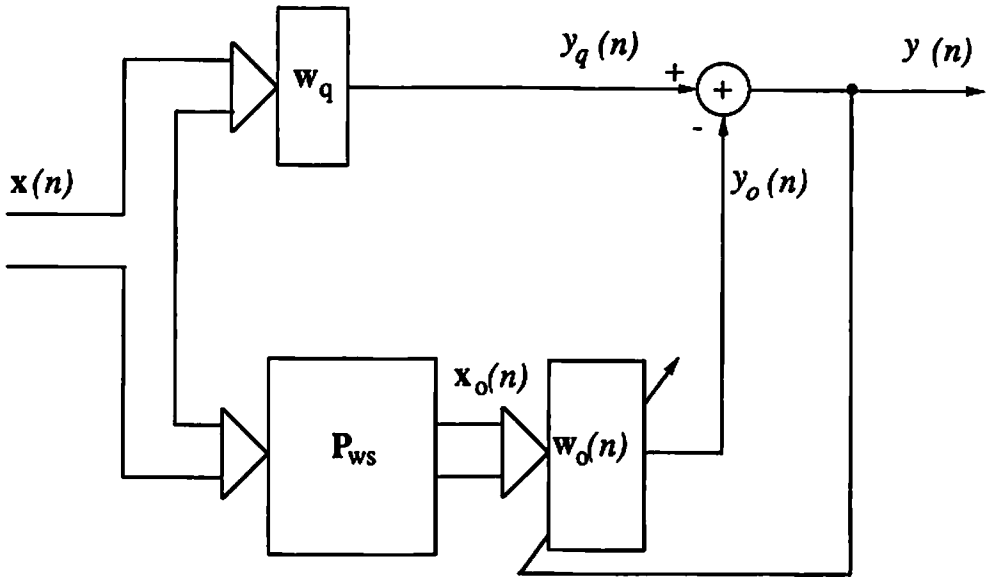

Figure 3.1: The block diagram of constrained-LMS

The function of the CLMS algorithm is characterized by the following equations:

$$\begin{aligned}
\mathbf{x}_o(n) &= \mathbf{P}_{ws}\mathbf{x}(n), \\
y_q(n) &= \mathbf{w}_q^{\dagger}\mathbf{x}(n), \\
y_o(n) &= \mathbf{w}_o^{\dagger}(n)\mathbf{x}_o(n), \\
y(n) &= y_q(n) - y_o(n), \\
\mathbf{w}_o(n+1) &= \mathbf{w}_o(n) + \mu y^*(n)\mathbf{x}_o(n) \\
\mathbf{w}(n) &= \mathbf{w}_q - \mathbf{w}_o(n)
\end{aligned} \tag{3.11}$$

And the block diagram of CLMS is shown in Figure 3.1.

Proof of Convergence

It was proven in [9] that the CLMS converges in mean under the condition of having the step size μ bounded by,

$$0 < \mu < 2/\lambda_{max} \tag{3.12}$$

where λ_{max} is the maximum eigenvalue of the matrix $\mathbf{P}_{ws}\mathbf{R}_{xx}\mathbf{P}_{ws}$.

3.2.2 The Generalized Sidelobe Canceller

The Algorithm

To illustrate the mechanism of GSC, the projection matrix $\mathbf{P}_{ws}$ expressed in terms of $\mathbf{W}_s$ is substituted into Eq.(3.5):

$$\mathbf{w}(n) = \mathbf{w}_q + \mathbf{W}_s(\mathbf{W}_s^\dagger \mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger \mathbf{w}(n). \tag{3.13}$$

This equation can be rewritten as

$$\mathbf{w}(n) = \mathbf{w}_q - \mathbf{W}_s \mathbf{w}_a(n) \tag{3.14}$$

by defining the $(N-M)$-dimensional weight vector

$$\mathbf{w}_a(n) = -(\mathbf{W}_s^\dagger \mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger \mathbf{w}(n), \tag{3.15}$$

which is the negative pseudo-inverse of $\mathbf{W}_s$ operating on $\mathbf{w}(n)$. The GSC [11] updates $\mathbf{w}_a(n)$ by unconstrained LMS adaptation,

$$\mathbf{w}_a(n+1) = \mathbf{w}_a(n) + \mu y^*(n)\mathbf{x}_a(n) \tag{3.16}$$

where

$$\mathbf{x}_a(n) = \mathbf{W}_s^\dagger \mathbf{x}(n) \tag{3.17}$$

is termed as the reduced-dimensional vector.

The function of the GSC algorithm is characterized by the following equations:

$$\begin{aligned}
\mathbf{x}_a(n) &= \mathbf{W}_s^\dagger \mathbf{x}(n), \\
y_q(n) &= \mathbf{w}_q^\dagger \mathbf{x}(n), \\
y_a(n) &= \mathbf{w}_a^\dagger(n)\mathbf{x}_a(n), \\
y(n) &= y_q(n) - y_a(n), \\
\mathbf{w}_a(n+1) &= \mathbf{w}_a(n) + \mu y^*(n)\mathbf{x}_a(n) \\
\mathbf{w}(n) &= \mathbf{w}_q - \mathbf{W}_s \mathbf{w}_a(n)
\end{aligned} \tag{3.18}$$

And the block diagram of GSC is shown in Figure 3.2.

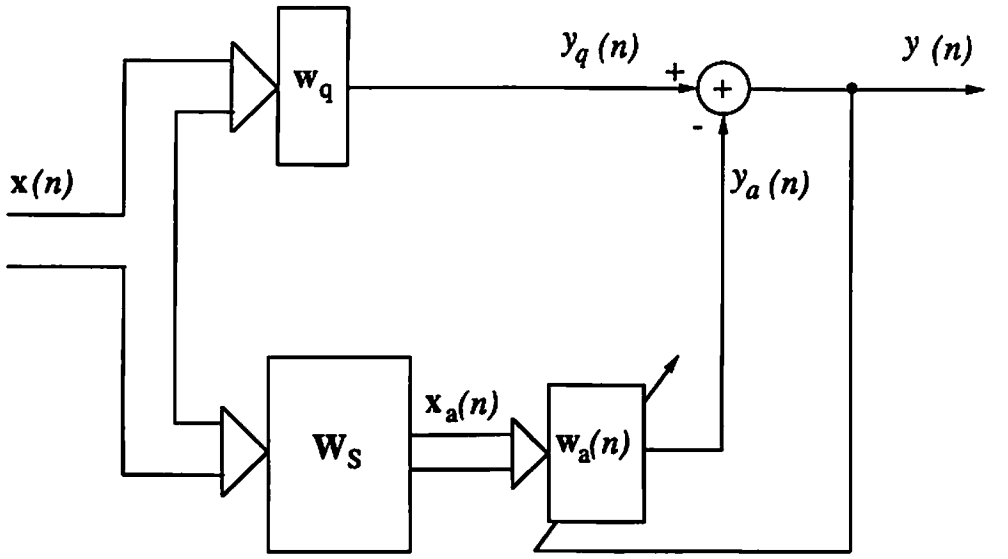

Figure 3.2: The block diagram of generalized sidelobe canceller

Proof of Convergence

To prove the GSC algorithm converges to the optimal solution, it is sufficient to prove that $\mathbf{w}_a(n)$ converges to

$$\mathbf{w}_{aopt} = (\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{w}_q \tag{3.19}$$

and use the fact (has already been proved in Section 2.4.2) that

$$\mathbf{w}_{opt} = \mathbf{w}_q - \mathbf{W}_s (\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s)^{-1} \mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{w}_q. \tag{3.20}$$

Referring to Figure 3.2, the weights $\mathbf{w}_a(n)$ in the lower path adaptively form a linear combination with the vector $\mathbf{x}_a(n)$ to produce a least-squares estimate of $y_q(n)$ in the upper path. Thus, with a properly chosen step size, the mean values of $\mathbf{w}_a(n)$ converge to the Weiner solution

$$\mathbf{w}_{aopt} = \mathbf{R}_{aa}^{-1} \mathbf{p}_{aq} \ , \tag{3.21}$$

where,

$$\begin{aligned} \mathbf{R}_{aa} &= E\{\mathbf{x}_a \mathbf{x}_a^\dagger\} \ , \\ &= \mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s \ , \end{aligned} \tag{3.22}$$

and,

$$\begin{aligned} \mathbf{p}_{aq} &= E\{\mathbf{x}_a y_q^\dagger\} , \\ &= \mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{w}_q . \end{aligned} \tag{3.23}$$

As a result, the convergence of the GSC to the optimal solution is granted under the condition of having the step size μ bounded by,

$$0 < \mu < 2/\lambda_{max} \tag{3.24}$$

where λ_{max} is the maximum eigenvalue of the matrix $\mathbf{W}_s^\dagger \mathbf{R}_{xx} \mathbf{W}_s$.

3.2.3 The Modified Generalized Sidelobe Canceller

The Algorithm

Given a constraint matrix $\mathbf{C}$ (assumed full rank), there exists at least one nonsingular matrix $\mathbf{A}$ which transforms $\mathbf{C}$ into $\hat{\mathbf{C}}$ with the following form:

$$\hat{\mathbf{C}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{V} \end{bmatrix} \tag{3.25}$$

where $\mathbf{V}$ is a $M \times M$ nonsingular matrix. The transformation is defined as

$$\hat{\mathbf{C}} = \mathbf{A}^\dagger \mathbf{C}. \tag{3.26}$$

and the matrix $\hat{\mathbf{C}}$ is termed as a canonical form. Using the matrix $\mathbf{A}$ we can reformulate the original constrained minimization problem into an equivalent problem with a constraint matrix of canonical form. First, note that $\mathbf{C}^\dagger \mathbf{w} = \mathbf{f}$ can be rewritten in terms of $\hat{\mathbf{C}}$ as

$$\hat{\mathbf{C}}^\dagger \hat{\mathbf{w}} = \mathbf{f} \tag{3.27}$$

by defining

$$\hat{\mathbf{w}} = \mathbf{A}^{-1} \mathbf{w}. \tag{3.28}$$

This corresponds to expressing the weights in new coordinates. Second, the system output power $\mathbf{w}^\dagger \mathbf{R}_{xx} \mathbf{w}$ can be expressed in terms of $\hat{\mathbf{w}}$ as

$$\hat{\mathbf{w}}^\dagger \hat{\mathbf{R}}_{xx} \hat{\mathbf{w}} \tag{3.29}$$

where

$$\hat{\mathbf{R}}_{xx} = \mathbf{A}^{\dagger}\mathbf{R}_{xx}\mathbf{A} \tag{3.30}$$

is the covariance matrix of the *transformed signal* defined as

$$\hat{\mathbf{x}} = \mathbf{A}^{\dagger}\mathbf{x}. \tag{3.31}$$

Thus, the original minimization problem is equivalent to a new minimization problem expressed in the new coordinates, which is,

$$\min_{\hat{\mathbf{w}}} \ \hat{\mathbf{w}}^{\dagger}\hat{\mathbf{R}}_{xx}\hat{\mathbf{w}}$$

subject to

$$\hat{\mathbf{C}}^{\dagger}\hat{\mathbf{w}} = \mathbf{f}.$$

The GSC weight decomposition can now be applied to solve this equivalent minimization problem. Since the upper $(N - M) \times M$ submatrix of $\hat{\mathbf{C}}$ is zero, the full rank matrix

$$\hat{\mathbf{W}}_s = \begin{bmatrix} \mathbf{I}_{(N-M)\times(N-M)} \\ \mathbf{0} \end{bmatrix} \tag{3.32}$$

clearly satisfies

$$\hat{\mathbf{W}}_s^{\dagger}\hat{\mathbf{C}} = \mathbf{0} \tag{3.33}$$

and is picked as the blocking matrix. (The term $\mathbf{I}$ is an identity matrix with its size as a subscript.) The quiescent weight vector in the new coordinate system is given by,

$$\hat{\mathbf{w}}_q = \hat{\mathbf{C}}(\hat{\mathbf{C}}^{\dagger}\hat{\mathbf{C}})^{-1}\mathbf{f} \tag{3.34}$$

which can be simplified as

$$\hat{\mathbf{w}}_q = \begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{w}}_{qm} \end{bmatrix}, \tag{3.35}$$

where

$$\hat{\mathbf{w}}_{qm} = (\mathbf{V}^{\dagger})^{-1}\mathbf{f}. \tag{3.36}$$

It should be noted that $\hat{\mathbf{w}}_q$ is the optimal solution when the transformed signal $\hat{\mathbf{x}}(n)$ has identity covariance matrix. The quiescent output for the MGSC is

$$\begin{aligned} \hat{y}_q(n) &= \hat{\mathbf{w}}_q^\dagger \hat{\mathbf{x}}(n) && (3.37) \\ &= \hat{\mathbf{w}}_{qm}^\dagger \hat{\mathbf{x}}_q(n) && (3.38) \end{aligned}$$

where $\hat{\mathbf{x}}_q(n)$ is the last M components of $\hat{\mathbf{x}}(n)$. The weights are updated by LMS algorithm as follows,

$$\hat{\mathbf{w}}_a(n+1) = \hat{\mathbf{w}}_a(n) + \mu y^*(n)\hat{\mathbf{x}}_a(n) \tag{3.39}$$

where

$$\begin{aligned} \hat{\mathbf{x}}_a(n) &= \hat{\mathbf{W}}_s^\dagger \hat{\mathbf{x}}(n) && (3.40) \\ y(n) &= \hat{y}_q(n) - \hat{\mathbf{w}}_a^\dagger(n)\hat{\mathbf{x}}_a(n). && (3.41) \end{aligned}$$

Note that $\hat{\mathbf{x}}_a(n)$ is the first $N-M$ components of $\hat{\mathbf{x}}(n)$.

The function of the MGSC is characterized by the following equations:

$$\begin{aligned} \hat{\mathbf{x}}(n) &= \mathbf{A}^\dagger \mathbf{x}(n) \\ &\triangleq \begin{bmatrix} \hat{\mathbf{x}}_a(n) \\ --- \\ \hat{\mathbf{x}}_q(n) \end{bmatrix}, \\ \hat{y}_q(n) &= \hat{\mathbf{w}}_{qm}^\dagger \hat{\mathbf{x}}_q(n), \\ \hat{y}_a(n) &= \hat{\mathbf{w}}_a^\dagger(n)\hat{\mathbf{x}}_a(n), \\ y(n) &= \hat{y}_q(n) - \hat{y}_a(n), \\ \hat{\mathbf{w}}_a(n+1) &= \hat{\mathbf{w}}_a(n) + \mu y^*(n)\hat{\mathbf{x}}_a(n) \\ \mathbf{w}(n) &= \mathbf{A}[\hat{\mathbf{w}}_q - \hat{\mathbf{W}}_s \hat{\mathbf{w}}_a(n)] \\ &= \mathbf{A} \cdot \begin{bmatrix} -\hat{\mathbf{w}}_a(n) \\ \hat{\mathbf{w}}_{qm} \end{bmatrix} \end{aligned} \tag{3.42}$$

And the block diagram of the MGSC is shown in Figure 3.3.

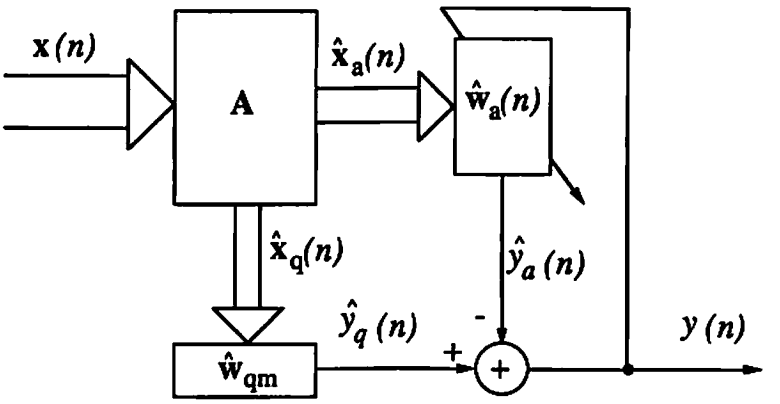

Figure 3.3: The block diagram of modified generalized sidelobe canceller

Proof of Convergence

In the MGSC algorithm, we first reformulate the original minimization problem into an equivalent problem and then apply the GSC algorithm. As a result, the convergence of MGSC immediately follows from that of the GSC. The step size μ in the MGSC has to satisfy the bound,

$$0 < \mu < 2/\lambda_{max} \tag{3.43}$$

where λ_{max} is the maximum eigenvalue of the matrix $\hat{\mathbf{W}}_s^\dagger \hat{\mathbf{R}}_{xx} \hat{\mathbf{W}}_s$.

3.3 Comparisons

3.3.1 Algebraic Comparison

Weight decomposition

Define a blocking matrix $\mathbf{W}_s$ such that

$$\mathbf{C}(\mathbf{C}^\dagger\mathbf{C})^{-1}\mathbf{C}^\dagger = \mathbf{I} - \mathbf{W}_s(\mathbf{W}_s^\dagger\mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger \tag{3.44}$$

then, at time n, the filter weights can be decomposed into two orthogonal complement subspaces as

$$\begin{aligned} \mathbf{w}(n) &= \mathbf{P}_c\mathbf{w}(n) + \mathbf{P}_{ws}\mathbf{w}(n) && (3.45)\\ &= \mathbf{C}(\mathbf{C}^\dagger\mathbf{C})^{-1}\mathbf{C}^\dagger\mathbf{w}(n) + \mathbf{P}_{ws}\mathbf{w}(n) && (3.46)\\ &= \mathbf{C}(\mathbf{C}^\dagger\mathbf{C})^{-1}\mathbf{f} + \mathbf{P}_{ws}\mathbf{w}(n) && (3.47)\\ &= \mathbf{w}_q + \mathbf{W}_s(\mathbf{W}_s^\dagger\mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger\mathbf{w}(n). && (3.48) \end{aligned}$$

The filter output $y(n) = \mathbf{w}^\dagger(n)\mathbf{x}(n)$ can be expressed as the difference of these two signals,

$$y(n) = y_q(n) - y_o(n) \tag{3.49}$$

by defining the quiescent signal $y_q(n)$ and the orthogonal signal $y_o(n)$ as

$$y_q(n) = \mathbf{w}_q^\dagger\mathbf{x}(n) \tag{3.50}$$

and

$$y_o(n) = -\mathbf{w}^\dagger(n)\mathbf{W}_s(\mathbf{W}_s^\dagger\mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger\mathbf{x}(n). \tag{3.51}$$

In the following, the CLMS, GSC and MGSC are compared by the way they generate these two signals.

The CLMS

The CLMS algorithm projects the filter weights $\mathbf{w}(n)$ and input data vector $\mathbf{x}(n)$ onto the constraint orthogonal space:

$$\begin{aligned} \mathbf{w}_o(n) &= -\ \mathbf{W}_s(\mathbf{W}_s^\dagger\mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger\mathbf{w}(n)\\ \mathbf{x}_o(n) &= \quad\ \mathbf{W}_s(\mathbf{W}_s^\dagger\mathbf{W}_s)^{-1}\mathbf{W}_s^\dagger\mathbf{x}(n). \end{aligned} \tag{3.52}$$

The weights $\mathbf{w}_o(n)$ are updated in a unconstrained fashion and the signal $y_o(n)$ is generated by the inner product of $\mathbf{w}_o(n)$ and $\mathbf{x}_o(n)$.

From CLMS to GSC

The GSC further simplifies the way to generate the orthogonal signal $y_o(n)$ by defining the reduced-dimensional weights and input data vector as

$$\begin{aligned} \mathbf{w}_a(n) &= -\ (\mathbf{W}_s\mathbf{W}_s^\dagger)^{-1}\mathbf{W}_s^\dagger\mathbf{w}(n) \\ \mathbf{x}_a(n) &= \quad \mathbf{W}_s^\dagger\mathbf{x}(n). \end{aligned} \tag{3.53}$$

The weights $\mathbf{w}_a(n)$ are updated in a unconstrained fashion and the signal $y_o(n)$ is generated by the inner product of $\mathbf{w}_a(n)$ and $\mathbf{x}_a(n)$.

In the CLMS, a weight vector of size N, $\mathbf{w}_o$, is used to control the filter weights projected onto a $N-M$ dimensional subspace, specified by $\mathbf{W}_s$. Thus, it in fact has some redundancy. This redundancy is removed in GSC by controlling the adaptive weights using only $N-M$ weights, $\mathbf{w}_a(n)$. Besides, the computation of $\mathbf{x}_a(n)$ in GSC is simplier than that of $\mathbf{x}_o(n)$ in CLMS since the multiplication of $\mathbf{x}(n)$ by the projection matrix $\mathbf{P}_{ws}$ is more complicated than that by the blocking matrix $\mathbf{W}_s$. This is because the uniqueness of a projection matrix restricts the effort of simplifying the computation $\mathbf{P}_{ws}\mathbf{x}(n)$ while the non-uniqueness of a vector space basis provides different approaches of implementing the computation $\mathbf{W}_s^\dagger\mathbf{x}(n)$.

From GSC to MGSC

It is easy to verify that the matrix $\mathbf{A}$ in the MGSC can always be constructed such that

$$\mathbf{A}^{-1}\mathbf{W}_s = \begin{bmatrix} \mathbf{I}_{(N-M)\times(N-M)} \\ \mathbf{0}_{M\times M} \end{bmatrix} \tag{3.54}$$

where $\mathbf{I}$ denotes the identity matrix with size indicated by its subscript. Using this matrix $\mathbf{A}$ the vector $\mathbf{A}^{-1}\mathbf{w}_q$ is computed and is partitioned into the upper $N-M$ and lower M components as:

$$\mathbf{A}^{-1}\mathbf{w}_q = \begin{bmatrix} \hat{\mathbf{w}}_b \\ --- \\ \hat{\mathbf{w}}_{qm} \end{bmatrix}. \tag{3.55}$$

Define a new weight vector $\hat{\mathbf{w}}(n)$ in terms of $\mathbf{A}^{-1}$ as

$$\hat{\mathbf{w}}(n) = \mathbf{A}^{-1}\mathbf{w}(n) \tag{3.56}$$

$$= \mathbf{A}^{-1}\mathbf{w}_q - \mathbf{A}^{-1}\mathbf{W}_s\mathbf{w}_a(n) \tag{3.57}$$

$$= \begin{bmatrix} \hat{\mathbf{w}}_b - \mathbf{w}_a(n) \\ \hat{\mathbf{w}}_{qm} \end{bmatrix}. \tag{3.58}$$

The negative of the first $N - M$ components of $\hat{\mathbf{w}}(n)$ are defined as the reduced-dimensional weights in the new coordinates, i.e.,

$$\hat{\mathbf{w}}_a(n) = -\hat{\mathbf{w}}_b + \mathbf{w}_a(n). \tag{3.59}$$

The filter output can then be expressed as

$$y(n) = \hat{\mathbf{w}}_{qm}^{\dagger}\hat{\mathbf{x}}_q(n) - \hat{\mathbf{w}}_a^{\dagger}(n)\hat{\mathbf{x}}_a(n) \tag{3.60}$$

by defining $\hat{\mathbf{x}}_a(n)$ and $\hat{\mathbf{x}}_q(n)$ be the first $N - M$ and the last M components of the transformed signal

$$\hat{\mathbf{x}}(n) = \mathbf{A}^{\dagger}\mathbf{x}(n). \tag{3.61}$$

In the MGSC, the matrix $\mathbf{A}$ essentially expresses the filter weights in a new coordinate system such that the upper $N - M$ weights, $\hat{\mathbf{w}}_a(\mathrm{n})$, are adaptive and the lower M weights, $\hat{\mathbf{w}}_{qm}$, are fixed. The filter output is formed by the inner product of the new weights $\hat{\mathbf{w}}(n)$ with the transformed signal $\hat{\mathbf{x}}(n)$. Compared to GSC, the quiescent weights in MGSC are *compressed* into M weights, $\hat{\mathbf{w}}_{qm}$, which further removes the redundancy in the GSC. In addition, the computations of $\mathbf{W}_s^{\dagger}\mathbf{x}(n)$ and $\mathbf{A}^{\dagger}\mathbf{x}(n)$ have the same complexity provided that $\mathbf{W}_s$ and $\mathbf{A}$ are constructed in decomposed forms [22]. The details of implementing $\mathbf{A}$ in decomposed form will be given in Chapter 4.

3.3.2 Geometrical Comparison

Geometrical representation

The function of an adaptive linearly-constrained filter of size $N = 2$ with one constraint, $M = 1$, can be visualized by the geometrical plot as shown in Figure 3.4. In

this plot, the coordinates with variables w_1 and w_2 correspond to the filter weights; the filter output power $\mathbf{w}^\dagger\mathbf{R}_{xx}\mathbf{w}$, which is a convex ball *sitting* on this w_1-w_2 plane, can be represented as contours of ellipsoidal shapes where each contour corresponds to a constant output power; the constraint equation $\mathbf{C}^\dagger\mathbf{w} = \mathbf{f}$ defines a straight line, l_1, on the plane; the constraint orthognal space defines a second straight line, l_2, passing through the origin and in parallel to l_1. All eligible weights, $\mathbf{w}(n)$, are restricted on the line l_1 since they all have to satisfy the constraint. The optimal weight vector $\mathbf{w}_{opt}$ is the intersection of l_1 and the contour with smallest output power; the quiescent weight vector $\mathbf{w}_q$, however, is the closest point on l_1 to the origin and perpendicular to the line l_2. Figure 3.4 can also be interpreted as in a general case by viewing l_1 as a M-dimensional hyperplane, l_2 as a $N-M$-dimensional subspace and each weight vector as a N-dimensional vector. Based on this geometrical plot, the behaviors of how the CLMS, GSC and MGSC algorithms move the current filter weights $\mathbf{w}(n)$ to the new weights $\mathbf{w}(n+1)$ are explained in the following.

CLMS

Referring to Figure 3.5, the CLMS first estimates a temporary new weight vector $\tilde{\mathbf{w}}(n+1)$ which, however, does not lie on l_1; the projection of $\tilde{\mathbf{w}}(n+1)$ onto l_2, denoted by $-\mathbf{w}_o(n+1)$,is then computed; since $\mathbf{w}_o(n+1)$ is on l_2, it is orthogonal to (perpendicular to) the constraint space, i.e., $\mathbf{C}^\dagger\mathbf{w}_o(n+1) = 0$; the actual new weights $\mathbf{w}(n+1)$ is obtained by subtracting $\mathbf{w}_o(n+1)$ from $\mathbf{w}_q$. The new weight vector obtained by the above procedures is ensured to lie on l_1 since subtracting a vector on l_2 from $\mathbf{w}_q$ will not bring it away from l_1.

GSC

Referring to Figure 3.6, the GSC estimates the weight vector $\mathbf{w}_o(n+1)$ directly on l_2 by expressing it as a linear combination of the column(s) of $\mathbf{W}_s$ with coefficient(s) designated by $\mathbf{w}_a(n+1)$; the current weight(s) $\mathbf{w}_a(n)$ on l_2 moves to the new weights $\mathbf{w}_a(n+1)$ by unconstrained LMS algorithm using the reduced-dimensional vector $\mathbf{x}_a(n) = \mathbf{W}_s^\dagger\mathbf{x}(n)$ as input data; the actual new weight vector $\mathbf{w}(n+1)$ is obtained

by subtracting $\mathbf{w}_o(n+1)$ from $\mathbf{w}_q$. Since the movements of the weight(s) are done on l_2, no constraint violation can occur.

MGSC

Referring to Figure 3.7, the MGSC first changes the coordinates w_1-w_2 into new coordinates $\hat{w}_1$-$\hat{w}_2$ by a transformation $\mathbf{A}$ such that $\hat{w}_1$ is align with $\mathbf{w}_q$ (in the plot, $\mathbf{A}$ is assumed orthogonal which corresponds to a rotation operation); the vector $\mathbf{w}_q$ can then be expressed in the new coordinates with only one (M in general) non-zero element; the GSC algorithm is applied to update weights in the new coordinate system.

3.4 Implementation and Analysis Models

3.4.1 A Unified Implementation Model

The CLMS, GSC and MGSC algorithms essentially reformulate a linearly constrained minimization problem into a unconstrained one with appropriate transformations. Using these reformulations, adaptive linearly-constrained filters are decoupled into fixed-weight and adaptive-weight portions as shown in Figure 3.8. The fixed-weight portion, depending upon the constraint equations, properly generates the reference signal $d(n)$ and data vector $\tilde{\mathbf{x}}(n)$; the adaptive-weight portion then adaptively forms a least-squares estimate of $d(n)$ by a linear combination of the vector $\tilde{\mathbf{x}}(n)$ with its weights. Thus, the adaptive-weight portion is basically a unconstrained adaptive filter with an available reference signal. The fixed-weight portion, however, can be characterized as a matrix transformation which transforms a constrained problem into a unconstrained one. The CLMS, GSC and MGSC actually state the property to which the matrix transformation has to obey. Detailed implementation issues using this model will be addressed in Chapter 4.

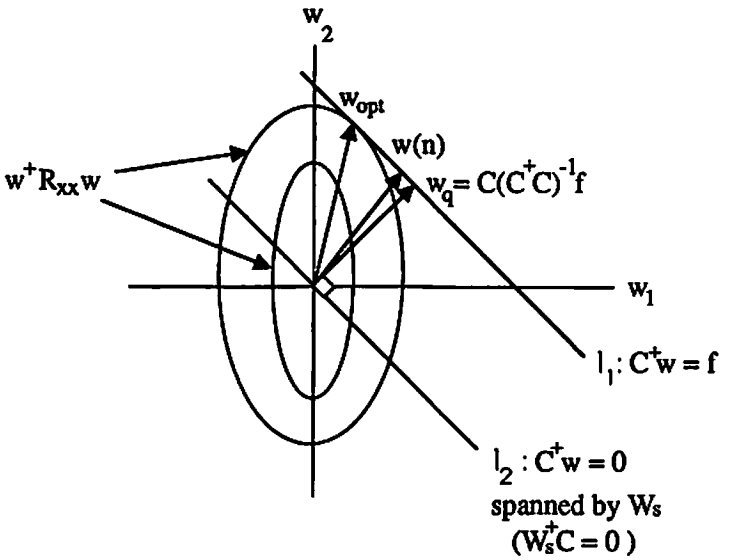

Figure 3.4: The conceptualized diagram of adaptive linearly-constrained filter.

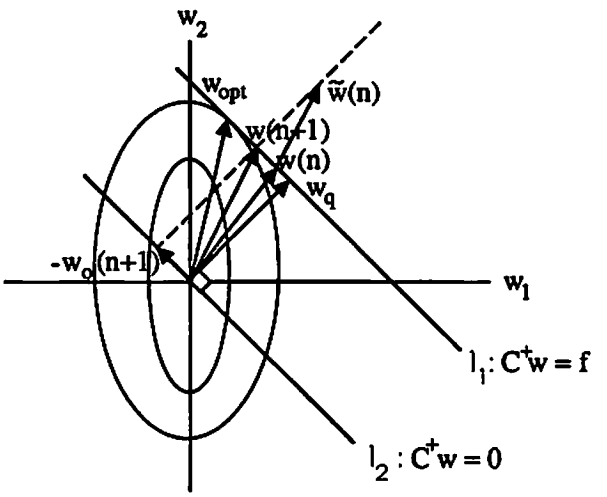

Figure 3.5: The conceptualized diagram of the CLMS algorithm.

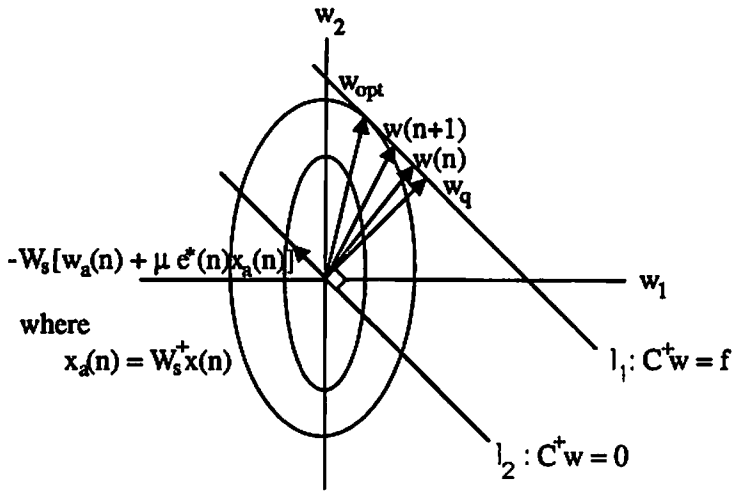

Figure 3.6: The conceptualized diagram of GSC algorithm.

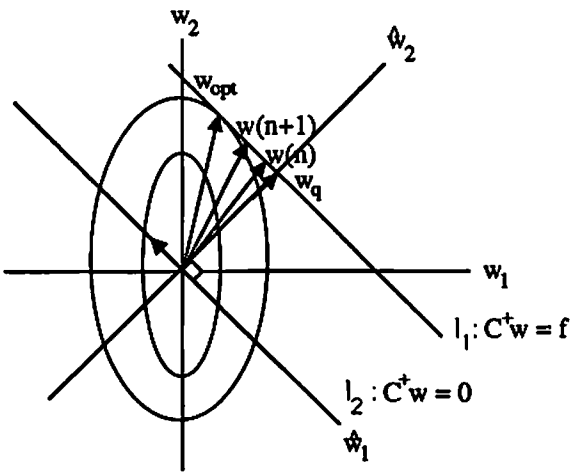

Figure 3.7: The conceptualized diagram of MGSC algorithm.

3.4.2 A Unified Analysis Model

Although the CLMS, GSC and MGSC all converge to the same optimal solution as shown previously, it is still not clear at this point which has the best performance in terms of convergence rate and misadjustment. It is shown in the following that these three algorithms are identical under certain conditions.

The identical conditions of CLMS and GSC

Theorem 3 *With the same step sizes, the CLMS and GSC are identical under the following two conditions:*

(a) *Initial condition:* $\mathbf{w}_o(0) = \mathbf{W}_s\mathbf{w}_a(0)$,

(b) *Orthogonal blocking matrix:* $\mathbf{W}_s^\dagger\mathbf{W}_s = \mathbf{I}$.

Proof:

From the weight decomposition mentioned earlier, condition (a) obviously implies that the CLMS and GSC start at the same initial weights.

In the following, we prove the CLMS and GSC are identical at time $n+1$ from the assumption that it is true at time n. Multiplying both sides of the LMS update equation in GSC (Eq.(3.16)) by $\mathbf{W}_s$, we have

$$\mathbf{W}_s\mathbf{w}_a(n+1) = \mathbf{W}_s\mathbf{w}_a(n) + \mu y^*(n)\mathbf{W}_s\mathbf{x}_a(n). \tag{3.62}$$

The identity of CLMS and GSC at time n implies

$$\mathbf{W}_s\mathbf{w}_a(n) = \mathbf{w}_o(n) \tag{3.63}$$

and the filter outputs are equal. On the other hand, $\mathbf{W}_s$ is orthogonal implies

$$\begin{aligned} \mathbf{W}_s\mathbf{x}_a(n) &= \mathbf{W}_s\mathbf{W}_s^\dagger\mathbf{x}(n) && (3.64)\\ &= \mathbf{P}_{ws}\mathbf{x}(n) && (3.65)\\ &= \mathbf{x}_o(n). && (3.66) \end{aligned}$$

As a result, Eq.(3.62) converts into

$$\mathbf{W}_s\mathbf{w}_a(n+1) = \mathbf{w}_o(n) + \mu y^*(n)\mathbf{x}_o(n) \tag{3.67}$$

$$= \mathbf{w}_o(n+1). \tag{3.68}$$

Thus, by induction, the CLMS and GSC are identical under conditions (a) and (b).

Q.E.D.

The identical conditions of GSC and MGSC

Theorem 4 *With the same step sizes, the GSC and MGSC are identical under the following two conditions:*

(a) *Initial condition:* $\hat{\mathbf{w}}_a(0) = -\hat{\mathbf{w}}_b + \mathbf{w}_a(0)$,

(b) *Similarity:* $\mathbf{A}\hat{\mathbf{W}}_s = \mathbf{W}_s$,

where $\hat{\mathbf{w}}_b$ *and* $\hat{\mathbf{W}}_s$ *are as defined in Eq.(3.55) and Eq.(3.32), respectively. Note that condition (b) implies that the first* $N - M$ *columns of* A *are equal to the columns of* $\mathbf{W}_s$.

Proof:

The initial overall filter weight vector in the MGSC is

$$\mathbf{A}\hat{\mathbf{w}}(0) = \mathbf{A} \cdot \begin{bmatrix} -\hat{\mathbf{w}}_a(0) \\ \hat{\mathbf{w}}_{qm} \end{bmatrix}. \tag{3.69}$$

Applying conditions (a) and (b) to above equation, it becomes

$$\mathbf{A}\hat{\mathbf{w}}(0) = \mathbf{A} \cdot \begin{bmatrix} \hat{\mathbf{w}}_b - \mathbf{w}_a(0) \\ \mathbf{w}_{qm} \end{bmatrix} \tag{3.70}$$

$$= \mathbf{A} \begin{bmatrix} \hat{\mathbf{w}}_b \\ \mathbf{w}_{qm} \end{bmatrix} - \mathbf{A}\hat{\mathbf{W}}_s\mathbf{w}_a(0) \tag{3.71}$$

$$= \mathbf{w}_q - \mathbf{W}_s\mathbf{w}_a(0). \tag{3.72}$$

This proves that the GSC and MGSC both start at the same initial weights.

In the following, we prove the GSC and MGSC are identical at time $n+1$ from the assumption that it is true at time n. At time $n+1$, the weight vector of MGSC is

$$\mathbf{A}\hat{\mathbf{w}}(n+1) = \mathbf{A}\hat{\mathbf{w}}_q - \mathbf{A}\hat{\mathbf{W}}_s\hat{\mathbf{w}}_a(n+1). \tag{3.73}$$

Since the first $N-M$ columns of $\mathbf{A}$ are equal to the columns of $\mathbf{W}_s$, it is true that

$$\hat{\mathbf{x}}_a(k) = \mathbf{x}_a(k) \tag{3.74}$$

for all time k. Applying this equality to Eq.(3.39), we have

$$\hat{\mathbf{w}}_a(n+1) = \hat{\mathbf{w}}_a(n) + \mu y^*(n)\mathbf{x}_a(n). \tag{3.75}$$

Substituting this equation into Eq(3.73) and using condition (b), we have,

$$\begin{aligned}
\mathbf{A}\hat{\mathbf{w}}(n+1) &= \mathbf{A}\hat{\mathbf{w}}_q - \mathbf{W}_s\hat{\mathbf{w}}_a(n) - \mu y^*(n)\mathbf{W}_s\mathbf{x}_a(n) && (3.76)\\
&= \mathbf{A}\hat{\mathbf{w}}_q - \mathbf{A}\hat{\mathbf{W}}_s\hat{\mathbf{w}}_a(n) - \mu y^*(n)\mathbf{W}_s\mathbf{x}_a(n) && (3.77)\\
&= \mathbf{A}\hat{\mathbf{w}}(n) - \mu y^*(n)\mathbf{W}_s\mathbf{x}_a(n) && (3.78)
\end{aligned}$$

The identity of GSC and MGSC at time n implies

$$\mathbf{A}\hat{\mathbf{w}}(n) = \mathbf{w}_q - \mathbf{W}_s\mathbf{w}_a(n) \tag{3.79}$$

and the filter outputs are equal. Using these two facts, Eq. (3.78) converts to

$$\begin{aligned}
\mathbf{A}\hat{\mathbf{w}}(n+1) &= \mathbf{w}_q - \mathbf{W}_s\mathbf{w}_a(n) - \mu y^*(n)\mathbf{W}_s\mathbf{x}_a(n) && (3.80)\\
&= \mathbf{w}_q - \mathbf{W}_s\mathbf{w}_a(n+1). && (3.81)
\end{aligned}$$

Thus, by induction, the MGSC and GSC are identical under conditions (a) and (b).

Q.E.D.

Summary

Since for a given $\mathbf{W}_s$ matrix in the GSC algorithm there always corresponds a matrix $\mathbf{A}$ in the MGSC such that these two algorithms are identical, the MGSC is actually a simpler implementation form for the GSC algorithm. On the other hand, the

CLMS algorithm is only a special case of the GSC algorithm when $\mathbf{W}_s$ is orthogonal. As a result, with different choices of $\mathbf{W}_s$ matrix the GSC algorithm represents the whole class of adaptive linearly-constrained filters. A model for analyzing the system performance by the use of different $\mathbf{W}_s$ matrix in the GSC algorithm is shown in Figure 3.9. Instead of directly varying the $\mathbf{W}_s$ matrix, an orthogonal $\mathbf{W}_s$ is first chosen and fixed while an additional nonsingular matrix $\mathbf{T}$ is used to transform one blocking matrix to the other. Consequently, the blocking matrix in this model is a product of two matrices $\mathbf{W}_s$ and $\mathbf{T}$. By properly choosing $\mathbf{T}$, the GSC algorithm with any blocking matrix can be represented by this model. This model is useful for the discussion of the effect of using several different blocking matrices in the GSC algorithm.

3.5 Summary

This chapter has presented three iterative algorithms for implementing the linearly constrained filter adaptively. Algebraic and geometric interpretations were used to illustrate the similarity of these three algorithms. In addition, it was shown that they are in fact identical under certain conditions. Two models were also established for implementation and analysis purposes.

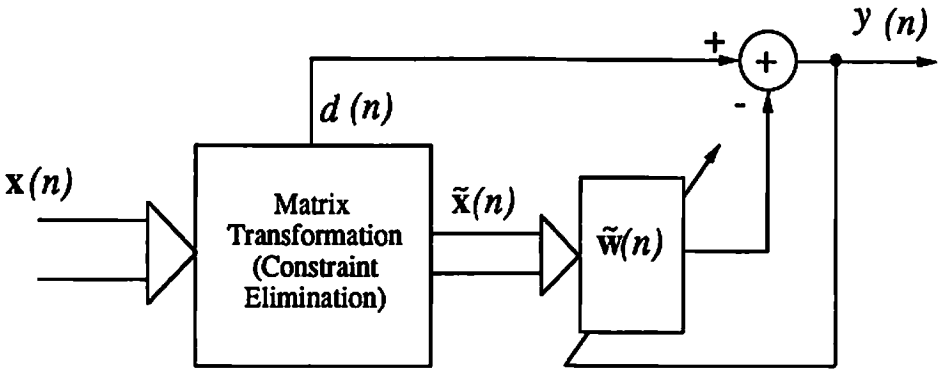

Figure 3.8: The unified implementation model

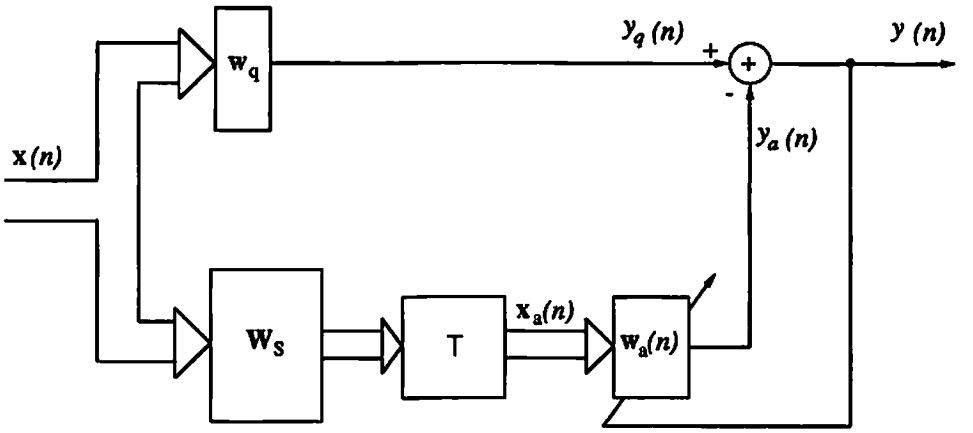

Figure 3.9: The unified analysis model

Chapter 4

Implementation of Adaptive Linearly-Constrained Filters

4.1 Introduction

This chapter focus on the issues of implementing the adaptive linearly-constrained filter using the implementation model as shown in last chapter where the filter weights are decoupled into fixed and adaptive weights. The adaptive-weight implementation is no different than the implementation of a conventional adaptive filter with desired signal, which has been well developed in the literature [24, 25]. Most material in this chapter is therefore devoted to the implementation of the fixed-weight portion.

Through the derivations of the iterative algorithms in last chapter the required property of the matrix the previous chapter used in the fixed-weight portion has been found. However, a systematic procedure is still needed to construct such a matrix transformation for implementing a particular linearly-constrained filter. Since the matrix transformation is not unique, it is preferable to construct a matrix transformation which has less computational complexity and good numerical behavior. To fully accomplish the efficiency of implementing the matrix transformation, two approaches are described to construct the matrix transformation in the case of small and large number of constraints, respectively.

In Section 4.2, the problem of implementing the matrix transformation is formulated based on the MGSC algorithm. Two structures, termed *the canonical form I and II*, are derived in Section 4.3 for implementing the matrix transformation. For a system with a small number of constraints (less than a half of the system size) the computation required in the canonical form I is less than that in the canonical form II and vice versa. Consequently, the option between canonical form I and II guarantees implementation efficiency for a system with either small or large number of constraints. The canonical forms are based on implementing the matrix transformation as a product of sparse matrices, which are referred to as *elementary reflectors.* Various applicable transformations for implementing the elementary reflectors are discussed in Section 4.4. A VLSI systolic array architecture for implementing the fixed-weight portion of the adaptive linearly-constrained filter is then proposed in Section 4.5. This chapter concludes with a comparison of this implementation to other related work.

4.2 Problem Formulation

In Section 3.4, a unified implementation model was presented where the filter weights were decomposed into fixed and adaptive portions. In the fixed-weight portion, the matrix transformation generate the reference signal $y_q(n)$ and the reduced-dimensional data vector $\mathbf{x}_a(n)$ from the input data vector $\mathbf{x}(n)$. Among the three iterative algorithms derived in the previous chapter the MGSC is the most efficient structure as discussed in Section 4.6. Thus, the matrix transformation is constructed in this chapter based on the MGSC weight decomposition. In the adaptive-weight portion, an adaptive filter with weights $\mathbf{w}_a(n)$ is used to form the least-mean-squares estimates of $y_q(n)$ by a linear combination of $\mathbf{x}_a(n)$. Although only LMS algorithm is applied to update the weights $\mathbf{w}_a(n)$ in this thesis, other adaptive algorithms (such as Least-squares, Q-R decomposition, *etc.*) can also be used. The use of an appropriate adaptive algorithm depends on the system performance required in a particular application. The system performance usually includes the considerations of **implementation cost,**

finite-wordlength effect, convergence rate and misadjustment. Detailed implementation of the fixed-weight portion of the adaptive linearly-constrained filter based on this model is discussed in the following sections.

4.3 Dual Implementation Structure

4.3.1 Canonical Form I

The A matrix

Let the constraint matrix C with M columns (full rank) be denoted as

$$\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2\ \cdots\ \mathbf{c}_M]. \tag{4.1}$$

The full rank property of C ensures that there exists at least one nonsingular matrix A such that

$$\mathbf{A}^\dagger\mathbf{C} = \begin{bmatrix} \mathbf{0}_{(N-M)\times M} \\ \mathbf{V}_{M\times M} \end{bmatrix} \tag{4.2}$$

where V is a $M \times M$ nonsingular matrix.

From the MGSC algorithm, the first $N - M$ components of $\mathbf{A}^\dagger\mathbf{x}(n)$ form the reduced-dimensional vector:

$$\mathbf{x}_a(n) = [\mathbf{A}^\dagger\mathbf{x}(n)]_{N-M,0} \tag{4.3}$$

and the inner product of the vector $\mathbf{w}_{qa}$ with the last M components of $\mathbf{A}^\dagger\mathbf{x}(n)$ forms the reference signal:

$$y_q(n) = \mathbf{w}_{qa}^\dagger[\mathbf{A}^\dagger\mathbf{x}(n)]_{0,M} \tag{4.4}$$

where

$$\mathbf{w}_{qa} = (\mathbf{V}^\dagger)^{-1}\mathbf{f}. \tag{4.5}$$

The A matrix and the $\mathbf{w}_{qa}$ vector then constitute the fixed-weight portion of the adaptive linearly-constrained filter with C and f specifying the constraint equations. This implementation structure is referred to as the canonical form I and is represented by the block diagram in Figure 4.1 (a).

Construction of the A matrix

We now derive a systematic procedure to construct the **A** matrix. It is constructed such that

$$\mathbf{A}^{\dagger}\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & \cdot & 0 & 0 \\ 0 & 0 & \cdot & \cdot & 0 & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & 0 & \times \\ \cdot & \cdot & 0 & \cdot & \times & \cdot \\ \cdot & 0 & \times & \cdot & \cdot & \cdot \\ 0 & \times & \cdot & \cdot & \times & \times \\ \times & \times & \times & \cdot & \times & \times \end{bmatrix} \tag{4.6}$$

where the j_th column of $\mathbf{A}^{\dagger}\mathbf{C}$ only has nonzero elements below the $(N-j)\_th$ row. The **V** matrix in Eq.(4.2) for this particular **A** therefore has only nonzero elements on or below its anti-diagonal. The matrices of this form are termed as the *lower anti-triangular matrices.*

For a constraint matrix **C** with M columns, the matrix **A** is decomposed as a product of M square submatrices:

$$\mathbf{A} = \mathbf{A}_1 \cdot \mathbf{A}_2 \cdots \mathbf{A}_M. \tag{4.7}$$

where the submatrices are used to successively transform the matrix **C**, column by column, to be a matrix of the form as shown in Eq.(4.6).

The procedure for constructing the **A** matrix in decomposed form then consists of M succeeding stages corresponding to the construction of the first submatrix $\mathbf{A}_1$ to the last submatrix $\mathbf{A}_M$. At the j_th stage of the procedure, the submatrix $\mathbf{A}_j$ is constructed to transform the first to the $(N-j)\_th$ elements of $\mathbf{c}_j$ to be zeros while remaining the rest elements of $\mathbf{c}_j$ unchanged; the matrix **C** is then updated by multiplying it by the Hermitian transpose of $\mathbf{A}_j$. The Gauss or Householder transformation which eliminates the elements of a vector can be used to construct the submatrix of **A** in each stage. The computational complexity using Gauss transformation is less than that of using Householder transformation. However, only using the Householder

transformation results in an orthogonal **A** matrix. Although self-orthogonal is not required for **A**, the 2-norm preservation property of an orthogonal matrix may improve numerical behavior in some cases. Comparisons of using orthogonal and unorthogonal **A** matrix in the adaptive linearly-constrained filtering will be discussed in later chapters.

The Gauss and Householder transformations are, respectively, the two simplest non-orthogonal and orthogonal transformations which can be used to construct the **A** matrix in decomposed form. However, incorporating these two transformations into the **A** matrix may not be suitable for parallel processing due to the need of global communications among the elements of $\mathbf{x}(n)$ while computing $\mathbf{A}^\dagger\mathbf{x}(n)$.

To accommodate to parallel processing, the Hermitian transpose of the j_th submatrix of **A** is further decomposed as a product of $N-j$ sparse matrices, that is,

$$\mathbf{A}_j^\dagger = \mathbf{A}_{N-j,j}\cdots\mathbf{A}_{2j}\cdot\mathbf{A}_{1j}, \tag{4.8}$$

where

$$\mathbf{A}_{ij} = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & & & & \cdot \\ \cdot & 0 & \times & \times & & \cdot \\ \cdot & \cdot & \times & \times & 0 & \cdot \\ \cdot & \cdot & & & 1 & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & 1 \end{bmatrix} \tag{4.9}$$

has ones on the diagonal and zeros elsewhere except that the elements on the entries $(i,i),(i,i+1),(i+1,i)$ and $(i+1,i+1)$ are chosen to force a zero at the (i,j) entry of **C**. The matrices of the form in the above equation are referred to as elementary reflectors. As the **A** matrix is decomposed as a sequence of elementary reflectors, computing $\mathbf{A}^\dagger\mathbf{x}(n)$ no longer needs global communication since multiplying any vector by an elementary reflector only needs two consecutive elements of that vector.

The procedure for constructing the **A** matrix as a sequence of elementary reflectors is summarized in the following algorithm.

Algorithm 1

For j = *1,2,..., M;*

For i = *1,2,...,N-j ;*

Find $\mathbf{A}_{ij}$ *such that the (i,j)_th entry of* $\mathbf{A}_{ij}\mathbf{C}$ *is zero;*

Update $\mathbf{C}$ *by* $\mathbf{A}_{ij}$, *i.e.,* $\mathbf{C} \leftarrow \mathbf{A}_{ij}\mathbf{C}$.

end;

end.

The selection of the elementary reflectors in the algorithm is discussed in the next section where various transformations are addressed.

The update of the C matrix in the algorithm results in a C matrix of the form as shown in Eq.(4.6). The bottom $M \times M$ of the final C corresponds to the V matrix needed for finding the $\mathbf{w}_{qa}$ vector according to Eq.(4.5).

Computation of $\mathbf{A}^{\dagger}\mathbf{x}(n)$

As A is in decomposed form, the transformed signal $\mathbf{A}^{\dagger}\mathbf{x}(n)$ can readily be computed via the M recursions,

$$\mathbf{x}^{(j)}(n) = \mathbf{A}_j^{\dagger}\mathbf{x}^{(j-1)}(n) \ , \tag{4.10}$$

with the initial condition $\mathbf{x}^{(0)}(n) = \mathbf{x}(n)$ and the final result $\mathbf{A}^{\dagger}\mathbf{x}(n) = \mathbf{x}^{(M)}(n)$. A conceptualized block diagram is shown in Figure 4.2 (a) to demonstrate this recursive computation. Further details on implementing it on a systolic VLSI architecture are discussed in Section 4.5. A reasonable question to ask at this point is what is the number of computations required to implement $\mathbf{A}^{\dagger}\mathbf{x}(n)$ when A is constructed by the decomposition process outlined above. The computational complexity is counted via the number of complex multiplications required since multiplier always represents the most costly component in hardware implementations. The total number of loops contained in the two nested loops in Algorithm 1 is

$$\kappa_a = N \times M - M(M+1)/2 \tag{4.11}$$

which designates the number of elementary reflectors in A as well. Since the multiplication of $\mathbf{x}(n)$ by $\mathbf{A}^{\dagger}$ is performed by successively multiplying $\mathbf{x}(n)$ by the elementary

reflectors, the total number of complex multiplications required in computing $\mathbf{A}^\dagger\mathbf{x}(n)$ is therefore proportional to κ_a, which is of order $O(NM)$. (The actual number of multiplications may vary by a constant factor depending on what type of transformation is applied to construct the elementary reflectors, as shown in the next section.) This decomposition approach is only efficient when the number of constraints, M, is small ($M \leq N/2$) since the number of computations grows proportional to M. In order to fully exploit the efficiency of implementing a system with arbitrary number of constraints, a second structure is developed in the following to deal with the case when the number of constraints is large.

4.3.2 Canonical Form II

The $\mathbf{B}^{-1}$ matrix

Besides the $\mathbf{A}$ matrix, the full rank property of $\mathbf{C}$ also ensures that there exists at least one nonsingular matrix $\mathbf{B}$ such that

$$(\mathbf{B}^{-1})^\dagger\mathbf{C} = \begin{bmatrix} \mathbf{U}_{M\times M} \\ \mathbf{0}_{(N-M)\times M} \end{bmatrix} \tag{4.12}$$

where $\mathbf{U}$ is a $M \times M$ nonsingular matrix.

As opposed to the $\mathbf{A}$ matrix, the $\mathbf{B}^{-1}$ matrix transforms the matrix $\mathbf{C}$ into a matrix form which has a non-singular $M \times M$ submatrix on the top instead of at the bottom. A proof similar to the proof of the MGSC algorithm can be derived to show that the last $N-M$ components of $(\mathbf{B}^{-1})^\dagger\mathbf{x}(n)$ form the reduced-dimensional vector:

$$\mathbf{x}_a(n) = [(\mathbf{B}^{-1})^\dagger\mathbf{x}(n)]_{0,N-M} \tag{4.13}$$

and the inner product of the vector $\mathbf{w}_{qb}$ with the first M components of $(\mathbf{B}^{-1})^\dagger\mathbf{x}(n)$ forms the reference signal:

$$y_q(n) = \mathbf{w}_{qb}^\dagger[(\mathbf{B}^{-1})^\dagger\mathbf{x}(n)]_{M,0} \tag{4.14}$$

where

$$\mathbf{w}_{qb} = (\mathbf{U}^\dagger)^{-1}\mathbf{f}. \tag{4.15}$$

The $\mathbf{B}^{-1}$ matrix and the $\mathbf{w}_{qb}$ vector then form the fixed-weight portion of the adaptive linearly-constrained filter with $\mathbf{C}$ and $\mathbf{f}$ specifying the constraint equations. This implementation structure is referred to as the canonical form II and is represented by the block diagram in Figure 4.1 (b).

Construction of the $\mathbf{B}^{-1}$ matrix

The decomposition process in the canonical form I cannot be applied to construct the $\mathbf{B}^{-1}$ matrix directly. Instead, the matrix $\mathbf{B}$ but not its inverse is first constructed in a decomposed form. The $\mathbf{B}^{-1}$ matrix is then obtained by inversing each individual submatrix of $\mathbf{B}$.

The property of $\mathbf{B}$ can be perceived from the constraint orthogonal space. Let a matrix $\tilde{\mathbf{W}}_s$ span the constraint orthogonal space and be denoted as

$$\tilde{\mathbf{W}}_s = [\mathbf{c}_{M+1}\, \mathbf{c}_{M+2} \cdots \mathbf{c}_N]. \tag{4.16}$$

Note that $\tilde{\mathbf{W}}_s$ is orthogonal to $\mathbf{C}$ and is full rank. The full rank property of $\tilde{\mathbf{W}}_s$ ensures that there exists at least one nonsingular matrix $\mathbf{B}$ such that

$$\mathbf{B}\tilde{\mathbf{W}}_s = \begin{bmatrix} \mathbf{0}_{M\times(N-M)} \\ \mathbf{Q}_{(N-M)\times(N-M)} \end{bmatrix} \tag{4.17}$$

where $\mathbf{Q}$ is a $(N-M)\times(N-M)$ nonsingular matrix. (We will prove that the inverse of this $\mathbf{B}$ matrix in fact results in Eq.(4.12).) In contrast to the $\mathbf{A}$ matrix in canonical form I, the $\mathbf{B}$ matrix transforms the blocking matrix, $\tilde{\mathbf{W}}_s$, instead of the constraint matrix, $\mathbf{C}$, into a matrix form having only a nonzero submatrix at the bottom. Since a nonsingular transformation will not change the rank of the matrix, the $\mathbf{Q}$ matrix must be of size $(N-M)\times(N-M)$ and nonsingular.

Multiplying both size of Eq.(4.17) by $\mathbf{B}^{-1}$, the matrix $\tilde{\mathbf{W}}_s$ can be expressed in terms of $\mathbf{B}^{-1}$ and $\mathbf{Q}$ as follows,

$$\tilde{\mathbf{W}}_s = \mathbf{B}^{-1} \cdot \begin{bmatrix} \mathbf{0}_{M\times(N-M)} \\ \mathbf{Q}_{(N-M)\times(N-M)} \end{bmatrix}. \tag{4.18}$$

Since the columns of $\tilde{\mathbf{W}}_s$ spans the constraint orthogonal space, we have

$$\tilde{\mathbf{W}}_s{}^{\dagger}\mathbf{C} = [\mathbf{0} \quad \mathbf{Q}^{\dagger}] \cdot (\mathbf{B}^{-1})^{\dagger}\mathbf{C} \tag{4.19}$$

$$= \mathbf{0}. \tag{4.20}$$

From the above equalities, the nonsingularity of $\mathbf{Q}$ implies that the lower $(N-M)\times M$ submatrix of $(\mathbf{B}^{-1})^{\dagger}\mathbf{C}$ must be zero and the full rank property of $\mathbf{C}$ implies that the upper $M \times M$ submatrix of $(\mathbf{B}^{-1})^{\dagger}\mathbf{C}$ must be nonsingular. As a result, the $(\mathbf{B}^{-1})^{\dagger}\mathbf{C}$ matrix must satisfy Eq.(4.12).

By comparing Eq.(4.2) and Eq.(4.17), it follows that the decomposition process for constructing the $\mathbf{A}$ matrix can also be applied to construct the $\mathbf{B}$ matrix since they all correspond to transforming a full rank matrix into a matrix having a zero submatrix on the top and a nonsingular submatrix at the bottom. However, a reference blocking matrix $\tilde{\mathbf{W}}_s$ must be established before applying the decomposition process. One approach to obtain $\tilde{\mathbf{W}}_s$ is to perform a singular value decomposition on $\mathbf{C}$ and pick the corresponding $N-M$ left singular vectors with zero singular values as its columns. Similar to the decomposition of $\mathbf{A}$, the matrix $\mathbf{B}$ is decomposed as a product of $N-M$ submatrices:

$$\mathbf{B} = \mathbf{B}_{N-M} \cdots \mathbf{B}_2 \cdot \mathbf{B}_1 \tag{4.21}$$

where the submatrices are used to successively transform the matrix $\tilde{\mathbf{W}}_s$, column by column, to be a matrix of the form as shown in Eq.(4.6) except that the size of the nonsingular submatrix at the bottom is now $(N-M)\times(N-M)$. The j_th submatrix $\mathbf{B}_j$ is used to force the j_th column of $\tilde{\mathbf{W}}_s$ to have only non-zero elements below the $(N-j)\_th$ row.

Similar to that of constructing the $\mathbf{A}$ matrix, the procedure of constructing the $\mathbf{B}$ matrix then consists of $N-M$ succeeding stages corresponding to the construction of the first submatrix $\mathbf{B}_1$ to the last submatrix $\mathbf{B}_{N-M}$. At each stage, a submatrix (either Gauss or Householder transformation) is constructed to introduce the zeros at the corresponding column of $\tilde{\mathbf{W}}_s$ and to update the $\tilde{\mathbf{W}}_s$ matrix.

The Hermitian transpose inverse of $\mathbf{B}$ in decomposed form is then obtained by

taking the Hermitian transpose inverse of each individual submatrix of $\mathbf{B}$, that is,

$$(\mathbf{B}^{-1})^{\dagger} = (\mathbf{B}_{N-M}^{-1})^{\dagger} \cdots (\mathbf{B}_{2}^{-1})^{\dagger} \cdot (\mathbf{B}_{1}^{-1})^{\dagger} \tag{4.22}$$

where the inverse operations are particularly simple for either Gauss or Householder transformations since only the operations of changing signs or taking Hermitian transpose are needed.

Similar to the decomposition of $\mathbf{A}_j^{\dagger}$ in canonical form I, the j_th submatrix of $\mathbf{B}$ can further be decomposed as a product of $N-j$ elementary reflectors:

$$\mathbf{B}_j = \mathbf{B}_{N-j,j} \cdots \mathbf{B}_{2j} \cdot \mathbf{B}_{1j}. \tag{4.23}$$

Correspondingly, the Hermitian transpose inverse of $\mathbf{B}_j$ is in a decomposed form

$$(\mathbf{B}_j{}^{-1})^{\dagger} = (\mathbf{B}_{N-j,j}{}^{-1})^{\dagger} \cdot (\mathbf{B}_{2j}{}^{-1})^{\dagger} \cdots (\mathbf{B}_{1j}{}^{-1})^{\dagger} \tag{4.24}$$

where the operation of inverting an elementary reflector only involves inverting a 2×2 matrix.

The procedure for constructing the $\mathbf{B}^{-1}$ matrix as a sequence of elementary reflectors is summarized in the following algorithm.

Algorithm 2

Perform a singular value decomposition on $\mathbf{C}$ *and*
pick the corresponding $N-M$ *left singular vectors*
with zero singular values as $\tilde{\mathbf{W}}_s$.
For j = 1,2,..., N-M;
    For i = 1,2,...,N-j ;
        Find $\mathbf{B}_{ij}$ *such that the (i,j)_th entry of* $\mathbf{B}_{ij}\tilde{\mathbf{W}}_s$ *is zero;*
        Update $\tilde{\mathbf{W}}_s$ *by* $\mathbf{B}_{ij}$, *i.e.,* $\tilde{\mathbf{W}}_s \leftarrow \mathbf{B}_{ij}\tilde{\mathbf{W}}_s$;
        Find the Hermitian transpose inverse of $\mathbf{B}_{ij}$;
        Update $\mathbf{C}$ *by* $(\mathbf{B}_{ij}^{-1})^{\dagger}$, *i.e.,* $\mathbf{C} \leftarrow (\mathbf{B}_{ij}^{-1})^{\dagger}\mathbf{C}$.
    end;

end.

The selection of the elementary reflectors in the algorithm is similar to that in Algorithm 1 and the Hermitian transpose inverse of an elementary reflector can be obtained from easily using the closed form formula of inverting a 2×2 matrix. More details on this will be discussed in Section 4.4.

The update of the **C** matrix is performed in the algorithm in order to find the $\mathbf{w}_{qb}$ vector. The top $M \times M$ matrix of the final **C** matrix corresponds to the **U** matrix which is used to solve for $\mathbf{w}_{qb}$ by Eq.(4.15).

Computation of $(\mathbf{B}^{-1})^{\dagger}\mathbf{x}(n)$

As $\mathbf{B}^{-1}$ is in decomposed form the transformed signal $(\mathbf{B}^{-1})^{\dagger}\mathbf{x}(n)$ can readily be computed via the $N - M$ recursions,

$$\mathbf{x}^{(j)}(n) = (\mathbf{B}_j{}^{-1})^{\dagger}\mathbf{x}^{(j-1)}(n) \ , \tag{4.25}$$

with the initial condition $\mathbf{x}^{(0)}(n) = \mathbf{x}(n)$ and the final result $(\mathbf{B}^{-1})^{\dagger}\mathbf{x}(n) = \mathbf{x}^{(N-M)}(n)$. A conceptualized block diagram is shown in Figure 4.2 (b) to demonstrate this recursive computation. Further details on implementing it on a systolic VLSI architecture is discussed in Section 4.5.

The total number of loops contained in the two nested loops in Algorithm 2 is

$$\kappa_b = N \times (N - M) - (N - M)(N - M + 1)/2 \tag{4.26}$$

which as well designates the number of elementary reflectors in $\mathbf{B}^{-1}$. The number κ_b is simply equal to κ_a by replacing M by $N - M$. Thus, the total number of complex multiplications in canonical form II is proportional to $N - M$ instead of M as in canonical form I. (The actual number of multiplications may vary by a constant factor depending on what type of transformation is applied to construct the elementary reflectors, as shown in the next section.) Therefore, canonical form II is more efficient than canonical form I for implementing a system with large number of constraints ($M > N/2$).

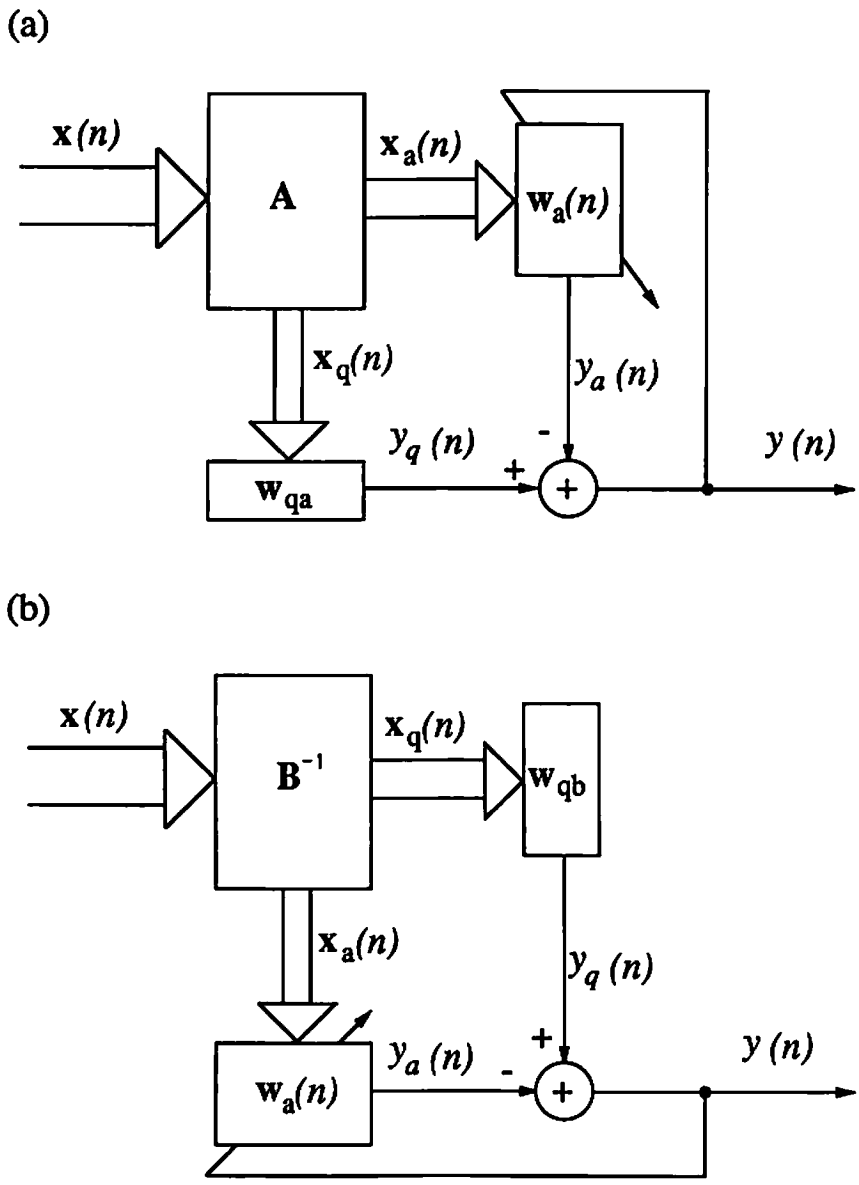

Figure 4.1: Two dual implementation structures:(a) canonical form I, (b) canonical form II.

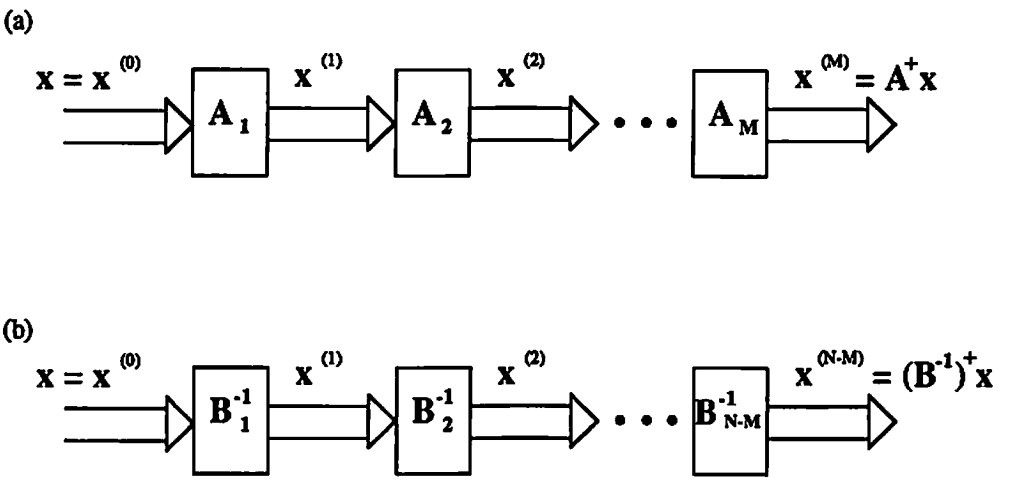

Figure 4.2: The decomposition of (a) the **A** matrix, (b) the $\mathbf{B}^{-1}$ matrix.

4.4 Incorporation of Transformations

In the last section, the construction of **A** (or **B**) was partitioned into a sequence of elementary reflectors in the form as shown in Eq.(4.9). Several applicable transformations on constructing such elementary reflectors are discussed in this section. Since each elementary reflector only eliminates one entry and affects two rows in the matrix **C** (or $\tilde{\mathbf{W}}_s$), it is sufficient to consider a 2×2 case. Thus, the problem of constructing an elementary reflector can be formulated as:

For a given vector

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \tag{4.27}$$

how to construct a non-singular transformation **T** to zero out the first element of **c**, i.e.,

$$\mathbf{T} \cdot \mathbf{c} = \begin{bmatrix} 0 \\ \hat{c}_2 \end{bmatrix} \tag{4.28}$$

where $\hat{c}_2$ is the update of the second element. Without loss of generality, two assumptions are made on the vector **c** to simplify the discussion of constructing the

transformation $\mathbf{T}$, which are, $|c_2| \geq |c_1|$ and $\mathbf{c} \neq \mathbf{0}$. The first assumption can always be obtained by properly exchanging c_1 and c_2 before constructing $\mathbf{T}$, which is equivalent to pre-multiplying a permutation matrix; the second assumption is, however, used to eliminate the trivial case when $\mathbf{c}$ equals zero. Three major categories in which $\mathbf{T}$ can belong to are: non-orthogonal, orthogonal and fast orthogonal transformations. In the following three subsections, the construction of $\mathbf{T}$ in these categories are discussed in details.

By using appropriate transformation, the computation $\mathbf{A}^t\mathbf{x}(n)$ in canonical form I can be implemented by a sequence of computations,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{4.29}$$

Similarly, the computation $(\mathbf{B}^{-1})^t\mathbf{x}(n)$ in canonical form II can be implemented by a sequence of computations,

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = (\mathbf{T}^{-1})^t \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{4.30}$$

We term these computations as the generic computations in the canonical form I and II, respectively. The computational complexity and numerical stability of the generic computations are two major factors which the designer has to determine what type of transformation should be used in his system.

4.4.1 Non-orthogonal Transformation

Gauss Transformation

The simplest transformation to eliminate the first element of the vector $\mathbf{c}$ is the Gauss transformation represented as

$$\mathbf{T} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \tag{4.31}$$

where $t = -c_1/c_2$. Note that the magnitude of t is bounded by unity which ensures numerical stability. The Hermitian transpose inverse of $\mathbf{T}$ is

$$(\mathbf{T}^{-1})^{\dagger} = \begin{bmatrix} 1 & 0 \\ -t^* & 1 \end{bmatrix} \tag{4.32}$$

where t^* is the complex conjugate of t. The generic computations are then represented as

$$\begin{cases} y_1 &= x_1 + t \cdot x_2 \\ y_2 &= x_2 \end{cases} \tag{4.33}$$

in canonical form I and

$$\begin{cases} y_1 &= x_1 \\ y_2 &= -t^* \cdot x_1 + x_2 \end{cases} \tag{4.34}$$

in canonical form II.

Gauss Transformation with Real Scale Factor

For fixed-point implementation, modification of the generic computations using Gauss transformation is necessary since numerical overflow may occur. This can be done by scaling y_1 by $1 + |t|$ in canonical form I and scaling y_2 by $1 + |t|$ in canonical form II. The modified generic computations are then equal to

$$\begin{cases} y_1 &= \frac{1}{1+|t|}x_1 + \frac{t}{1+|t|}x_2 \\ y_2 &= x_2 \end{cases} \tag{4.35}$$

in canonical form I and

$$\begin{cases} y_1 &= x_1 \\ y_2 &= \frac{-t^*}{1+|t|}x_1 + \frac{1}{1+|t|}x_2 \end{cases} \tag{4.36}$$

in canonical form II.

This modification, however, increases the computational complexity by 50% since an additional real-complex multiplier is needed for each generic computation. The additional cost for implementing the modified generic computations is substantially high, especially for hardware based implementation. An alternative modification is therefore derived using a complex scale factor.

Gauss Transformation with Complex Scale Factor

To simplify the derivation we only discuss the case in canonical form I, a similar complex scale factor can be applied to that in canonical form II. A complex scaling term s can be applied to y_1 in Eq.(4.33) as follows,

$$y_1 = (\frac{1}{s})x_1 + (\frac{t}{s})x_2 \tag{4.37}$$

where s is chosen as,

$$s = 1 + \text{csgn}(t) \cdot t \; . \tag{4.38}$$

In this expression, the complex function csgn(z) operating on a complex variable $z = z_R + jz_I$ is defined as,

$$\text{csgn}(z) = \begin{cases} \text{sgn}(z_R) & \text{if } |z_R| \geq |z_I| \\ -j\text{sgn}(z_I) & \text{otherwise} \end{cases} \tag{4.39}$$

and sgn($\cdot$) is the signum function for real variables. Note that the range of csgn(z) is $\{-1, 1, -j, j\}$ and that the complex term s is a generalization of the real scale factor $1 + |t|$ since they are equal when t is real.

With the use of Eq.(4.38), and a little algebra, the modified generic computation in Eq.(4.37) can be rewritten as,

$$y_1 \;=\; x_1 - \alpha \cdot \{x_1 - \sigma x_2\} \; , \tag{4.40}$$

where,

$$\sigma = \text{csgn}^* \; (t) \tag{4.41}$$

and

$$\alpha = \frac{\text{csgn}(t)t}{1 + \text{csgn}(t)t}. \tag{4.42}$$

In the implementation of (4.40), a wordlength of 2 bits is sufficient to encode σ since it can only be one of the four elements in the set {-1,1,$-j$,j} and a simple circuit with this control input can be used to implement the computation σx_1 without multiplication. (An example implementation is presented in the next section.) The

magnitude of the coefficient α can be bounded by $\frac{1}{2}$, and,in addition, the real part of α is always nonnegative. Because of this limitation, α has a smaller dynamic range than does t and is more efficiently encoded for a given word length.

When scaling of this type is employed, the growth factor for Eq.(4.40) is

$$K = \frac{1+|t|}{|s|} \tag{4.43}$$

$$= \frac{1+|t|}{|1+\mathrm{csgn}(t)t|}\,, \tag{4.44}$$

and satisfies the following inequalities,

$$1 \leq K \leq (\frac{2\sqrt{2}}{1+\sqrt{2}})^{\frac{1}{2}} \approx 1.08 \tag{4.45}$$

The lower bound is attained when t is real and the upper bound is reached when $t = \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$. In order to specify the efficiency of the scaled operation, it is important to understand how the term K varies with t. This can be achieved from an examination of the following function:

$$f(z) = \frac{1+|z|}{|1+\mathrm{csgn}(z)z|} \tag{4.46}$$

In polar coordinates,

$$r = |z|$$

$$\theta = \begin{cases} \arctan(z_I/z_R) & \text{if } |z_R| \geq |z_I| \\ \arctan(z_R/z_I) & \text{otherwise} \end{cases}$$

where $0 \leq r \leq 1$ and $-\frac{\pi}{4} \leq \theta \leq \frac{\pi}{4}$. Equation (4.46) can then be rewritten in a simpler form,

$$g(r,\theta) = \left(\frac{1+2r+r^2}{1+2r|\cos\theta|+r^2}\right)^{\frac{1}{2}}\,. \tag{4.47}$$

Figure 4.3 (a) and (b) illustrates the behavior as a function of r and $|\theta|$. It is an increasing function of r when $|\theta|$ is fixed and also increases with $|\theta|$ for fixed values of r. As a result, both the amplitude and phase of t determine the value of the growth factor in Eq.(4.44). The smoothness of these plots also suggests that this scaling technique has good performance for most cases.

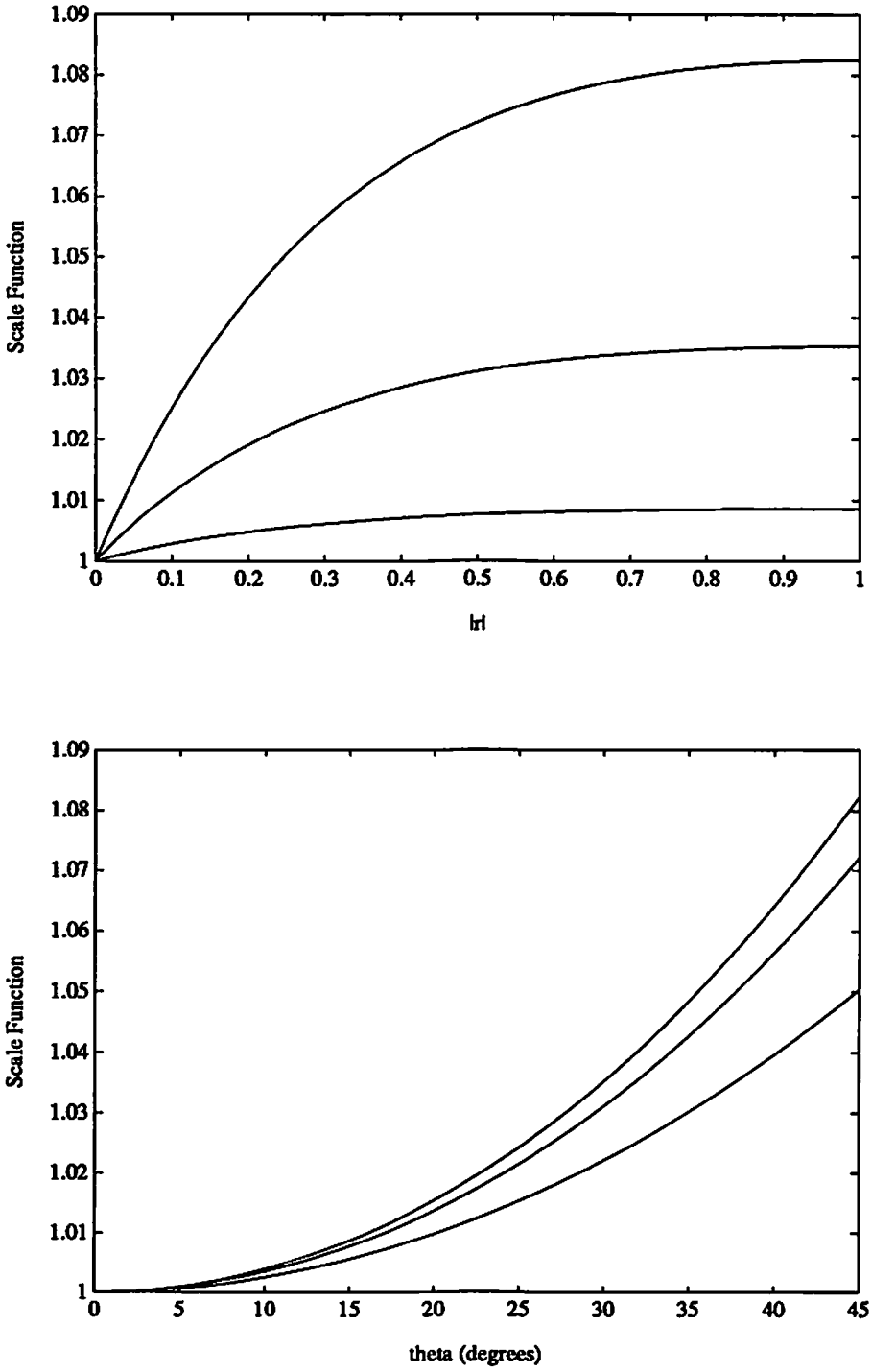

Figure 4.3: The growth factor analysis plots for complex scale factor.

Use of this scaling technique results in a reduction on the bound of the growth factor of a generic computation from 2 to 1.08 which substantially reduces the possibility of numerical overflow. In addition, the cost of incorporating this scaling technique is considerably lower.

4.4.2 Orthogonal Transformation

Orthogonal transformation $\mathbf{T}$ can be formed using the Givens rotation:

$$\mathbf{T} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{4.48}$$

where

$$c = \frac{1}{\sqrt{1+t^2}}, \quad s = \frac{t}{\sqrt{1+t^2}}, \quad \text{and} \quad t = -c_1/c_2. \tag{4.49}$$

Since the Hermitian transpose inverse of an orthogonal matrix is equal to the matrix itself, the generic computations in both canonical form I and II can all be represented as:

$$\begin{cases} y_1 = c\ x_1 + s x_2 \\ y_2 = -s\ x_1 + c x_2 \end{cases} \tag{4.50}$$

By ensuring the input vector has 2-norm bounded by unity, no numerical overflow will occur in the generic computations because orthogonal transformations do not change the 2-norm of a vector.

4.4.3 Fast Orthogonal Transformation

Concept of Fast Orthogonal Transformation

Suppose a matrix $\mathbf{A}$ is decomposed as a product of M submatrices:

$$\mathbf{A} = \mathbf{A}_1 \cdot \mathbf{A}_2 \cdots \mathbf{A}_M. \tag{4.51}$$

A sufficient condition to make $\mathbf{A}$ an orthogonal matrix is that all submatrices in the product are orthogonal, i.e.,

$$\mathbf{A}_j^{\dagger}\mathbf{A}_j = \mathbf{I}, \qquad \text{for} \quad j = 1, 2, \cdots, M. \tag{4.52}$$

A more general way to construct an orthogonal matrix is to require that all $\mathbf{A}_j$'s satisfying the recursions

$$\mathbf{A}_j^{\dagger}\mathbf{D}_{j-1}\mathbf{A}_j = \mathbf{D}_j, \qquad \text{for} \quad j = 1, 2, \cdots M \tag{4.53}$$

where each $\mathbf{D}_j$ is a diagonal matrix with the initial condition $\mathbf{D}_0 = \mathbf{I}$, as an identity matrix. The resulting matrix $\mathbf{A} \cdot \mathbf{D}_M^{-1/2}$ is then an orthogonal matrix. In this case the orthogonality is not required for each submatrix and the transformation is termed the Fast Orthogonal Transformation (FOT). The FOT was originally proposed in [26] to avoid the square root computations in solving least-squares problems, and it was termed the square-root free orthogonal transformation.

The FOT can be applied to construct an orthogonal $\mathbf{T}$ with less computation than the Givens Rotation. In the following, two different forms of FOT are derived to implement the transformation $\mathbf{T}$.

Fast Givens Transformation

Let the vector $\mathbf{c} = [c_1\ c_2]^T$ and the matrix $\mathbf{D} = diag(d_1, d_2)$ be given, where $d_1, d_2 > 0$, and define

$$\mathbf{T} = \begin{bmatrix} 1 & a_1 \\ b_1 & 1 \end{bmatrix}. \tag{4.54}$$

Observe that if

$$a_1 = -c_1/c_2 \quad \text{and} \quad b_1 = -(d_2/d_1)a_1^*, \tag{4.55}$$

then $\mathbf{T}$ satisfies Eq.(4.28) and

$$\mathbf{T}\mathbf{D}\mathbf{T}^{\dagger} = \begin{bmatrix} (1+r_1)d_1 & 0 \\ 0 & (1+r_1)d_2 \end{bmatrix} \tag{4.56}$$

where

$$r_1 = -a_1 b_1 = (d_2/d_1)|c_1/c_2|^2. \tag{4.57}$$

Similarly, we can define

$$\mathbf{T} = \begin{bmatrix} a_2 & 1 \\ 1 & b_2 \end{bmatrix} \tag{4.58}$$

and

$$a_2 = -c_2/c_1, \quad b_2 = -(d_1/d_2)a_2^*, \tag{4.59}$$

then $\mathbf{T}$ satisfies Eq.(4.28) and

$$\mathbf{TDT}^\dagger = \begin{bmatrix} (1+r_2)d_2 & 0 \\ 0 & (1+r_2)d_1 \end{bmatrix} \tag{4.60}$$

where

$$r_2 = -a_2 b_2 = (d_1/d_2)|c_2/c_1|^2. \tag{4.61}$$

Since $r_1 r_2 = 1$, we can always select $\mathbf{T}$ as either Eq.(4.54) or (4.58) so that the "growth factor" $(1 + r_i)$ is bounded by 2. Matrices of the form

$$\begin{bmatrix} 1 & a_1 \\ b_1 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} a_2 & 1 \\ 1 & b_2 \end{bmatrix} \tag{4.62}$$

are referred to as *fast Givens transformations*. Detailed discussion of the fast Givens transformation can be found in [19]. The Hermitian transpose inverse of Eq.(4.54) and (4.58) can be easily obtained as:

$$(\mathbf{T}^{-1})^\dagger = \frac{1}{|1 - a_1 b_1|} \begin{bmatrix} 1 & -b_1^* \\ -a_1^* & 1 \end{bmatrix} \tag{4.63}$$

and

$$(\mathbf{T}^{-1})^\dagger = \frac{1}{|1 - a_2 b_2|} \begin{bmatrix} b_2^* & 1 \\ 1 & a_2^* \end{bmatrix} \tag{4.64}$$

respectively. With properly pre-permuting the input (x_1 and x_2), and post-permuting the output (y_1 and y_2), the generic computations in canonical form I and II can be represented as:

$$\begin{cases} y_1 &= x_1 + b x_2 \\ y_2 &= a x_1 + x_2 \end{cases} \tag{4.65}$$

where a and b are coefficients of complex values.

Notice that the generic computation implemented by a fast Givens transformation required only about half of the multiplications as that by a Givens rotation. However, scaling may be necessary due to the possible overflow at each stage. The

scaling can be done similar to the non-orthogonal transformation as discussed in Section 4.4.1.

A Novel Fast Orthogonal Transformation

In addition to the fast Givens transformations, the following transformation can also be used as a fast orthogonal transformation. Define

$$\mathbf{T} = \begin{bmatrix} -b_1 & r_1 \\ b_1 & 1 \end{bmatrix} \tag{4.66}$$

where

$$a_1 = -c_1/c_2, \quad b_1 = -(d_2/d_1)a_1^*, \quad r_1 = (d_2/d_1)|a_1|^2 \tag{4.67}$$

We prove that this $\mathbf{T}$ can be used as the elementary reflector by showing that

$$\mathbf{Tc} = \begin{bmatrix} 0 \\ c_2(1+r_1) \end{bmatrix} \tag{4.68}$$

and

$$\mathbf{TDT}^\dagger = \begin{bmatrix} d_2 r_1(1+r_1) & 0 \\ 0 & d_2(1+r_1) \end{bmatrix}. \tag{4.69}$$

By direct computation, we have

$$\mathbf{Tc} = \begin{bmatrix} -b_1 c_1 + r_1 c_2 \\ b_1 c_1 + c_2 \end{bmatrix} \tag{4.70}$$

Since

$$\begin{aligned} -b_1 c_1 + r_1 c_2 &= \frac{d_2}{d_1} a_1^* \cdot c_1 + \frac{d_2}{d_1}|a_1|^2 c_2 && (4.71) \\ &= c_2 \frac{d_2}{d_1}[a_1^* \cdot (c_1/c_2) + |a_1|^2] && (4.72) \\ &= c_2 \frac{d_2}{d_1}[a_1^* \cdot (-a_1) + |a_1|^2] && (4.73) \\ &= 0, && (4.74) \end{aligned}$$

and

$$-b_1 c_1 + c_2 = -\frac{d_2}{d_1} a_1^* \cdot c_1 + c_2 \tag{4.75}$$

$$= c_2[-\frac{d_2}{d_1}a_1^* \cdot (c_1/c_2) + 1] \tag{4.76}$$

$$= c_2(\frac{d_2}{d_1}|a_1|^2 + 1) \tag{4.77}$$

$$= c_2(1 + r_1) \tag{4.78}$$

it follows that Eq. (4.68) is true. We also have

$$\mathbf{TDT}^\dagger = \begin{bmatrix} |b_1|^2 d_1 + r_1^2 d_2 & -|b_1|^2 d_1 + r_1 d_2 \\ -|b_1|^2 d_1 + r_1 d_2 & |b_1|^2 d_1 + d_2 \end{bmatrix}. \tag{4.79}$$

Evaluating the elements in the above matrix, we have the following equalities:

$$-|b_1|^2 d_1 + r_1 d_2 = -(\frac{d_2}{d_1})^2 |a_1|^2 d_1 + \frac{d_2}{d_1}|a_1|^2 d_2 \tag{4.80}$$

$$= -\frac{{d_2}^2}{d_1}|a_1|^2 + \frac{{d_2}^2}{d_1}|a_1|^2 \tag{4.81}$$

$$= 0, \tag{4.82}$$

$$|b_1|^2 d_1 + r_1^2 d_2 = (\frac{d_2}{d_1})^2 |a_1|^2 d_1 + (\frac{d_2}{d_1})^2 |a_1|^4 d_2 \tag{4.83}$$

$$= d_2[\frac{d_2}{d_1}|a_1|^2 + (\frac{d_2}{d_1})^2 |a_1|^4] \tag{4.84}$$

$$= d_2(r_1 + r_1^2) = d_2 r_1(1 + r_1) \tag{4.85}$$

and

$$|b_1|^2 d_1 + d_2 = (\frac{d_2}{d_1})^2 |a_1|^2 d_1 + d_2 \tag{4.86}$$

$$= d_2(\frac{d_2}{d_1}|a_1|^2 + 1) \tag{4.87}$$

$$= d_2(1 + r_1). \tag{4.88}$$

Therefore, $\mathbf{T}$ satisfies Eq.(4.69).

Similarly, we can also define

$$\mathbf{T} = \begin{bmatrix} r_2 & -b_2 \\ 1 & b_2 \end{bmatrix} \tag{4.89}$$

where

$$a_2 = -c_2/c_1, \quad b_2 = -(d_1/d_2)a_2^*, \quad r_2 = (d_1/d_2)|c_2/c_1|^2. \tag{4.90}$$

Following a similar proof as above, it can be shown that

$$\mathbf{Tc} = \begin{bmatrix} 0 \\ c_1(1+r_2) \end{bmatrix} \tag{4.91}$$

and

$$\mathbf{TDT}^\dagger = \begin{bmatrix} d_1 r_2(1+r_2) & 0 \\ 0 & d_1(1+r_2) \end{bmatrix}. \tag{4.92}$$

The Hermitian transpose inverse of Eq.(4.66) and (4.89) can easily be computed as

$$(\mathbf{T}^{-1})^\dagger = \frac{1}{|b_1(1+r_1)|} \begin{bmatrix} 1 & -b_1^* \\ -r_1 & -b_1^* \end{bmatrix} \tag{4.93}$$

and

$$(\mathbf{T}^{-1})^\dagger = \frac{1}{|b_2(1+r_2)|} \begin{bmatrix} b_2^* & -1 \\ b_2^* & r_2 \end{bmatrix} \tag{4.94}$$

respectively. With properly pre-permuting the input (x_1 and x_2), and post-permulting the output (y_1 and y_2), the generic computations in canonical form I and II can be represented as:

$$\begin{cases} y_1 = bx_1 + rx_2 \\ y_2 = -bx_1 \pm x_2 \end{cases} \tag{4.95}$$

where b and r are coefficients. The coefficient b is of complex value, however, r is always a real positive number. The generic computation using this fast orthogonal transformation requires less computation than that using the Fast Givens Transformation. This is due to the fact that two coefficients in the latter transformation are complex while one coefficient is complex and the other is real in the former transformation.

4.5 Systolic VLSI Architecture

A systolic VLSI planar array architecture is derived in this section to implement the fixed-weight portion of the linearly constrained filter based on the construction of **A** in decomposed form as addressed in Section 4.3.

The partition of the matrix $\mathbf{A}$ into a sequence of elementary reflectors naturally leads to a planar array architecture which corresponds to mapping each submatrix $\mathbf{A}_{ij}$ onto a Processing Element (PE). Figure 4.4 illustrates the detailed structure of the planar array implementation. The elements labeled with a $l\Delta$ symbol indicate time delays with l representing the number of units of delay. A total of $(N+M-3)$ delay elements and $N \times M - M(M+1)/2$ PE's are interconnected in a two-dimensional array. The PE's form a trapezoidal network in which the number of PE's reduces uniformly from $(N-1)$ at the first column to $(N-M)$ at the last column. Each PE has two inputs and two outputs and all PE's are identical except for internal coefficient values. Each PE implements one generic computation as defined in Eq.(4.29) in which the elementary reflector can be a Gauss transformation or Givens Rotation.

In the planar array, each PE communicates to its four neighbors by receiving the input data from its northern and western neighbors and transmitting the output data to its eastern and southern neighbors, respectively. The function of the PE at node (i,j) can be generally represented as in Figure 4.5 (a). The timing of the circuit is arranged such that the inputs arrive simultaneously and the outputs are generated after a one-unit time delay Δ, representing the time required for one internal computation of the PE. A unit-delay element in this path is therefore required to ensure that the horizontal and vertical inputs of each PE arrive at the same time. One convenient method for achieving coincident timing is to insert a one-unit time delay at the vertical input to the first PE of every column (except the first column), and to use increasing delays at the horizontal inputs to all PE's after the second, as shown in Figure 4.4. This approach guarantees that data will propagate from the input of the planar array to the input of PE at node (i,j) after a (2j+i-3)-unit time delay.

Input data samples from the vector $\mathbf{x}(n)$ are piped into the planar array in parallel with a time latency between successive vectors of one delay unit Δ. The delay elements which precede the PE's serve to skew the input data and produce a skewed data flow propagating through the array. On one hand, the first $N-M$ components of $\mathbf{A}^{\dagger}\mathbf{x}(n)$, denoted by $\mathbf{x}_a(n)$, first appear at the horizontal output after

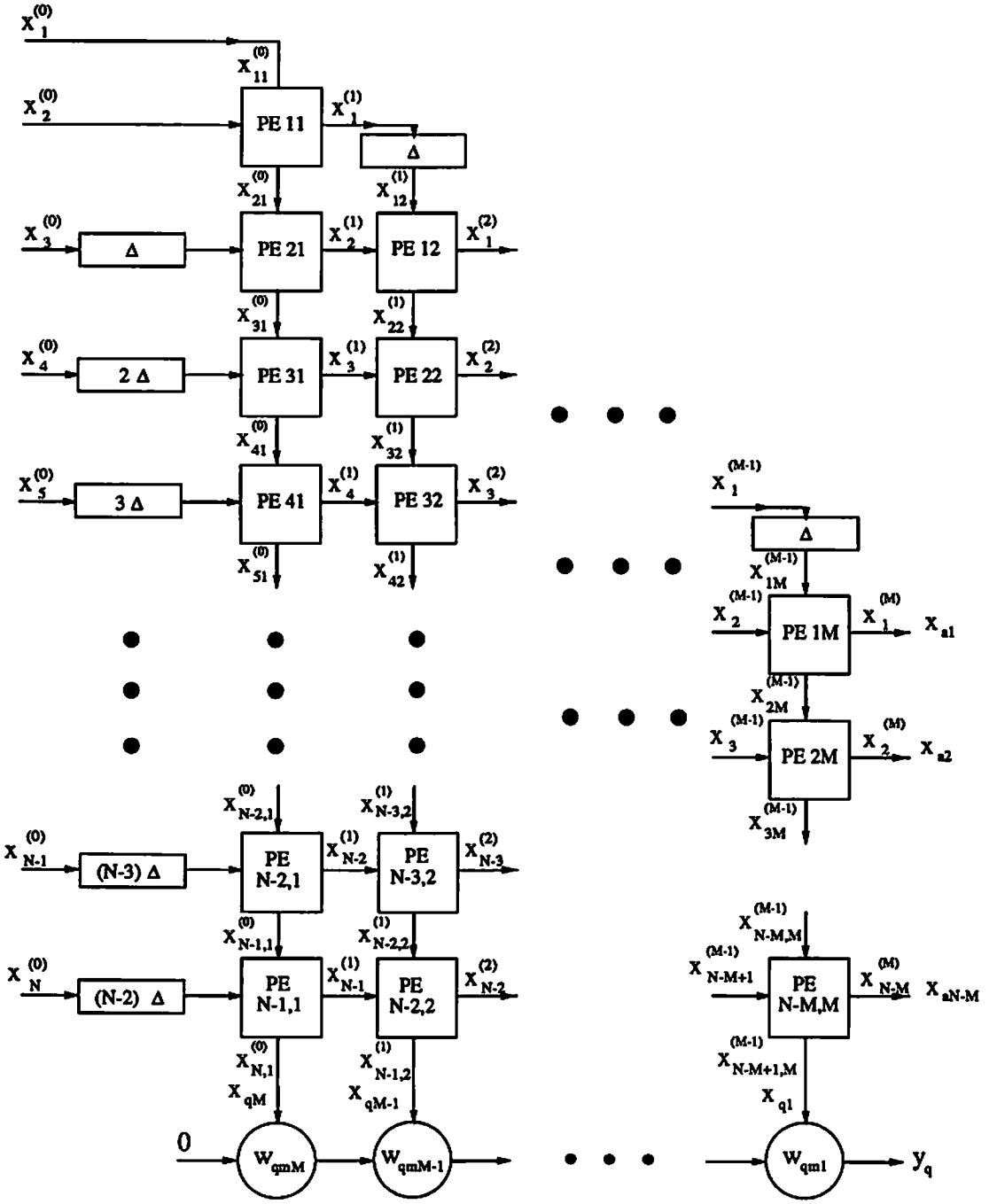

Figure 4.4: The VLSI planar array architecture.

a time delay of $(2M-1)$-unit but they do not all appear simultaneously. The first element of $\mathbf{x}_a(n)$ is produced at the output after $2M-1$ unit delays, the second appears after $2M$ units, and the last after $M+N-1$ units. This skewed output vector is followed exactly one time unit later by the next skewed output vector $\mathbf{x}_a(n+1)$. On the other hand, the last M components of $\mathbf{A}^\dagger\mathbf{x}(n)$, denoted by $\mathbf{x}_q(n)$, appear at the vertical output after a time delay of $(N-1)$-unit in a skewed order similar to the horizontal output. However, they appear in a reverse order such that the last element appears at the first column of the PE, the second last element appears at the second column, and the first at the last column. M boundary nodes are then used to compute the inner product of $\mathbf{w}_{qm}$ and $\mathbf{x}_q(n)$ to produce the desired signal $y_q(n)$ as shown in Figure 4.4. The function of the boundary node is represented in Figure 4.5 (b). By requiring the computation of each boundary node be done in one unit time delay Δ, the desired signal $y_q(n)$ appears at the output one-unit time delay after the last element of $\mathbf{x}_a(n)$. Once $\mathbf{x}_a(n)$ and $y_q(n)$ are generated, the adaptive-weight portion of the linearly constrained filter can be implemented using any unconstrained adaptive algorithm.

4.6 Comparisons to Previous Work

In addition to the canonical form I and II presented above, several approaches [27, 28, 29] have been proposed to implement the fixed-weight processor in the adaptive linearly-constrained filters. In the following, the previous approaches are reviewed and compared.

4.6.1 Previous Work Review

McWhirter and Shepherd

Define a matrix consisting of the constraint matrix $\mathbf{C}$ and the response vector $\mathbf{f}$ as follows:

$$\left[\begin{array}{cc} \mathbf{C}^\dagger & \mathbf{f} \end{array}\right]. \tag{4.96}$$

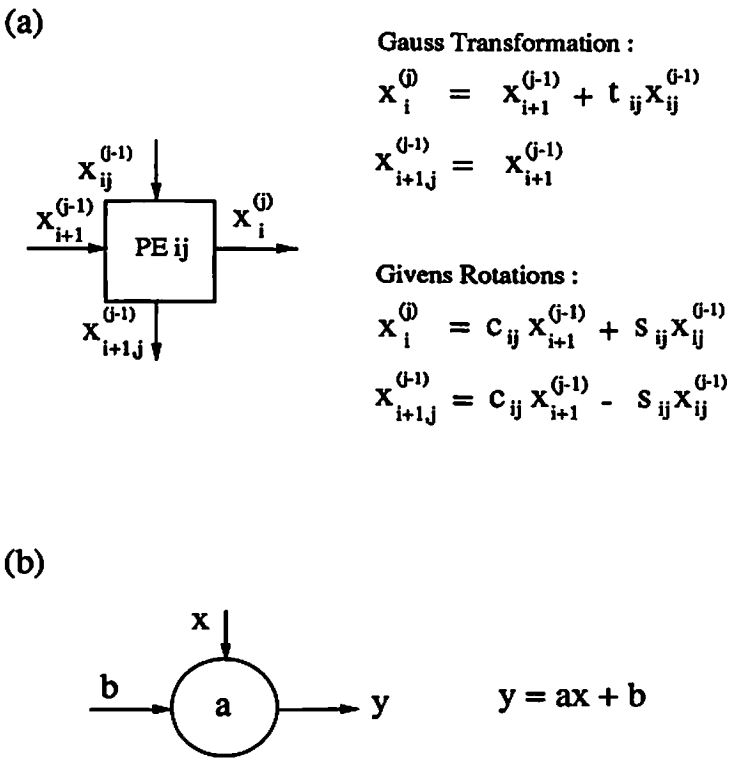

Figure 4.5: The functional diagram of the (a) internal PEs and (b) boundary nodes in the VLSI planar array.

Premultiply the above matrix by a unitary matrix $\mathbf{Q}$ and a diagonal matrix $\mathbf{D}$ to transfer the first $M \times M$ submatrix of $\mathbf{C}^\dagger$ into a triangular form, we have

$$\begin{bmatrix} \mathbf{T} & \mathbf{V} & \hat{\mathbf{f}} \end{bmatrix} = \mathbf{DQ} \begin{bmatrix} \mathbf{C}^\dagger & \mathbf{f} \end{bmatrix} \tag{4.97}$$

where $\mathbf{T}$ is a $M \times M$ upper triangular matrix, $\mathbf{V}$ is a $M \times (N-M)$ matrix and $\hat{\mathbf{f}}$ is a $M \times 1$ vector. Assuming the matrix $\mathbf{T}$ is invertible, the above matrix is further multiplied by $\mathbf{T}^{-1}$ which results in

$$\begin{bmatrix} \mathbf{I}_{M\times M} & \mathbf{T}^{-1}\mathbf{V} & \mathbf{T}^{-1}\hat{\mathbf{f}} \end{bmatrix}. \tag{4.98}$$

Accordingly, the original constraint equations $\mathbf{C}^\dagger\mathbf{w} = \mathbf{f}$ can be rewritten as

$$\tilde{\mathbf{C}}^\dagger \mathbf{w} = \tilde{\mathbf{f}} \tag{4.99}$$

where

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{I}_{M\times M} \\ (\mathbf{T}^{-1}\mathbf{V})^\dagger \end{bmatrix} \tag{4.100}$$

and

$$\tilde{\mathbf{f}} = \mathbf{T}^{-1}\hat{\mathbf{f}}. \tag{4.101}$$

Defining a nonsingular matrix $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{M\times M} & -\mathbf{T}^{-1}\mathbf{V} \\ \mathbf{0}_{(N-M)\times M} & \mathbf{I}_{(N-M)\times(N-M)} \end{bmatrix}, \tag{4.102}$$

it is easy to verify that

$$\mathbf{A}^\dagger\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{I}_{M\times M} \\ \mathbf{0}_{(N-M)\times M} \end{bmatrix}. \tag{4.103}$$

This $\mathbf{A}$ matrix then transforms the constraint matrix $\tilde{\mathbf{C}}$ into a form which has an identity matrix of size $M \times M$ on the top and zeros elsewhere. From the MGSC algorithm, the vectors $\mathbf{x}_q(n)$ and $\mathbf{x}_a(n)$ are, respectively, the first M and the last $N-M$ components of $\mathbf{A}^\dagger\mathbf{x}(n)$. The $\mathbf{w}_{qm}$ vector required for generating the reference signal is $y_q(n)$ equal to $\tilde{\mathbf{f}}$ since the nonsingular submatrix in $\mathbf{A}^\dagger\tilde{\mathbf{C}}$ is identity. Partitioning

the vector $\mathbf{x}(n)$ into two vectors corresponding the upper M lower $N-M$ components as follows

$$\mathbf{x}(n) = \begin{bmatrix} \mathbf{x}_u(n) \\ \mathbf{x}_l(n) \end{bmatrix}, \tag{4.104}$$

the reduced-dimensional vector $\mathbf{x}_a(n)$ and the reference signal $y_q(n)$ can then be formed by the following equations

$$\mathbf{x}_a(n) = \mathbf{x}_l(n) - \mathbf{V}^\dagger(\mathbf{T}^{-1})^\dagger \mathbf{x}_u(n) \tag{4.105}$$

$$y_q(n) = \hat{\mathbf{f}}^\dagger(\mathbf{T}^{-1})^\dagger \mathbf{x}_u(n). \tag{4.106}$$

These two equations are used in [27] to transform the constrained problem into a unconstrained one. The approach in [27] is therefore a special case of the MGSC algorithm using the $\mathbf{A}$ matrix defined in Eq.(4.102).

Van Veen and Roberts

Two different pre-processors with and without data permutations were proposed in [28] to implement the blocking matrix $\mathbf{W}_s$ of the GSC structure in the case when the number of constraints is large. The ideas of these two processors are summarized in the following.

(a) Pre-processor with data permutation

Let an arbitrary blocking matrix $\tilde{\mathbf{W}}_s$ of size $N \times (N-M)$ be given and be partitioned into the upper $(N-M) \times (N-M)$ and the lower $M \times (N-M)$ submatrices, i.e.,

$$\tilde{\mathbf{W}}_s = \begin{bmatrix} \mathbf{W}_{su} \\ \mathbf{W}_{sl} \end{bmatrix} \tag{4.107}$$

Row permutations are first used to ensure the upper submatrix $\mathbf{W}_{su}$ is nonsingular and a blocking matrix is defined by postmultiplying $\tilde{\mathbf{W}}_s$ by $\mathbf{W}_{su}^{-1}$, which is,

$$\mathbf{W}_s = \begin{bmatrix} \mathbf{I}_{(N-M)\times(N-M)} \\ \mathbf{W}_{sl}\mathbf{W}_{su}^{-1} \end{bmatrix}. \tag{4.108}$$

This $\mathbf{W}_s$ matrix satisfies the properties of a blocking matrix since postmultiplying $\tilde{\mathbf{W}}_s$ by a nonsingular matrix will not change its rank and its column space.

(b)Pre-processor without data permutation

The initial blocking matrix $\tilde{\mathbf{W}}_s$ can be decomposed into a product of two matrices **U** and **S**, i.e.,

$$\tilde{\mathbf{W}}_s = \mathbf{US} \tag{4.109}$$

where **U** is a $N \times (N-M)$ full rank matrix and **S** is a $(N-M) \times (N-M)$ nonsingular matrix. The decomposition is done such that **U** is of lower triangular form. By this partitioning the matrix **U** has the same column space and can therefore be used as the blocking matrix in the GSC structure, that is,

$$\mathbf{W}_s = \mathbf{U}. \tag{4.110}$$

This $\mathbf{W}_s$ matrix is simplier than $\tilde{\mathbf{W}}_s$ since it is of triangular form.

Kalson and Yao

The approach in [29] simply rederived the GSC structure in a one constraint case. The blocking matrix $\mathbf{W}_s$ is a full matrix of size $N \times (N-M)$ without any special structure.

4.6.2 Comparisons

A similar systolic array has been proposed in [27] for fixed-weight portion implementation in the linearly-constrained filter. In that approach, the constraint equations were reformulated such that the constraint matrix is partitioned into two submatrices with a $M \times M$ triangular matrix on the top and a $(N-M) \times M$ rectangular matrix at the bottom. The authors then showed that these two submatrices can be used to eliminate the constraints. As shown in earlier that this approach is actually a special case of the MGSC algorithm in which the **A** matrix was implemented as a block triangular matrix. The main drawback of this implementation is that it requires nonsingular principal submatrix assumption on **C** otherwise a data permutation pre-processor is needed. For an arbitrary given constraint matrix, the nonsingularity assumption is by no means guaranteed. As a result, a permutation pre-processor is generally needed.

It therefore complicates the implementation. In addition, no modification was available to implement the **A** matrix as an orthogonal matrix. Although self-orthogonal property is not required for **A**, it is sometimes desirable to have an orthogonal **A** to obtain better numerical stability in the system. Moreover, this approach is inefficient for the implementation of a large number of constraints.

Two systolic array architectures were presented in [28] to efficiently implement the large number of constraints cases. This approach was based on deriving a special structure for the blocking matrix $\mathbf{W}_s$ in the GSC algorithm such that the computational complexity is minimized. A reference blocking matrix $\tilde{\mathbf{W}}_s$ of arbitrary form was first constructed. Transformations were then used to transform $\tilde{\mathbf{W}}_s$ into a simpler form. In the first architecture, the blocking matrix is reduced to have an identity principal submatrix which, however, requires the assumption of nonsingular principal submatrix of $\tilde{\mathbf{W}}_s$. To implement this architecture therefore needs data permutation as required in [27]. The second architecture was then proposed to avoid the need of data permutation by transforming $\tilde{\mathbf{W}}_s$ into a lower triangular matrix.

Another systolic array was presented in [29] which implemented the $\mathbf{W}_s$ as a full matrix. No special structure was imposed on $\mathbf{W}_s$ to reduce computation.

As a summary, Table 4.6.2 is used to outline the characteristic of various approaches specified under the first column of the table. Under the second column, the characteristic of each individual matrix transformation are listed. The third column shows the number of complex multiplications needed to implement the fixed weight portion of the filter with the corresponding approach. The number of complex multiplications is expressed as the sum of two terms which correspond to the required computations for the desired signal and the reduced-dimensional vector, respectively. The variables κ_a and κ_b are as defined in Eq.(4.11) and (4.26), and the constant factor c varying from 1 to 3 depends upon the transformation used to construct the elementary reflectors. The last column of the table shows the required assumption on the constraint matrix **C** or the reference blocking matrix $\tilde{\mathbf{W}}_s$ in the corresponding approach.

Compared to other systolic arrays, the systolic array presented in this Chapter

Approach	Matrix Transformation	No. of Complex Multiplications	Assumption on C or $\tilde{\mathbf{W}}_s$
Canonical Form I	**A**: A sequence of elementary reflectors	$M + c \cdot \kappa_a$ $1 \leq c \leq 3$	None
Canonical Form II	$\mathbf{B}^{-1}$: A sequence of elementary reflectors	$M + c \cdot \kappa_b$ $1 \leq c \leq 3$	None
McWhirter and Shepherd	**A**: Upper block triangular	$M + \kappa_a$	Nonsingular principal submatrix of C
Van Veen and Roberts (a)	$\mathbf{W}_s$: Identity principal submatrix	$N + M \times (N - M)$	Nonsingular principal submatrix of $\tilde{\mathbf{W}}_s$
Van Veen and Roberts (b)	$\mathbf{W}_s$: Lower triangular	$N + \kappa_b$	None
Kalson and Yao	$\mathbf{W}_s$: Full matrix	$N + N \times (N - M)$	None

Table 4.1: The comparison table.

possess at least three advantages : 1) No nonsingular assumption on the principal submatrix of C or $\tilde{\mathbf{W}}_s$ avoids the need of data permutation pre-processor, 2) Various transformations for constructing the elementary reflectors provides more flexibility for implementation, 3) The choice between canonical form I and II ensures efficiency for both small and large numbers of constraints.

4.7 Summary

This chapter has presented a systematic procedure to implement the adaptive linearly-constrained filter. The approach was based on using the MGSC algorithm to decouple the filter weights into fixed and adaptive portions in which the former is completely determined by the constraint equations while the latter is adaptively formed to minimize the output power. In general, the adaptive-weight portion can be implemented

by some well-developed adaptive algorithms such as recursive least-squares or a simple LMS approach. Furthermore, with the use of matrix decomposition techniques, it was shown that the fixed-weight portion can be implemented efficiently for an arbitrary number of constraints.

The design methodologies presented by other authors [27, 28, 29] in developing the systolic implementations of the linearly-constrained filter were also based on the fixed and adaptive weights decoupling. In fact, their approaches on implementing the fixed-weight portion are equivalent to implementing the GSC blocking matrix or the MGSC transformation matrix as mentioned in Section 4.5. However, the implementations in their approaches were in direct form in which the blocking matrix and the transformation matrix were formed explicitly. In contrast to the direct form implementations, the decomposed form implementation used in this paper does not form the transformation matrix explicitly. Instead, only the factorization of that matrix is implemented. The use of decomposed form in this paper is similar to that of a widely used approach in solving linear system or least-squares problems in which the original problem is reformulated as an equivalent and simple one by a sequence of matrix transformations. As such it provides favorable numerical property with the use of the Gauss transformation, the Householder transformation or the Givens rotation. In addition, the flexibility offered by the choices among these transformations also allows the system designer a trade-off between implementation cost and numerical performance. This trade-off flexibility is especially important in special purpose hardware implementations in which the hardware cost is a critical consideration.

Chapter 5

Fixed-Point Implementation

5.1 Introduction

The implementation of signal processing algorithms on special-purpose hardware systems is practically important because of its potential for high speed processing. The arithmetic being implemented can either be fixed-point or floating-point depending upon the particular application. These two arithmetics have different features. Floating point introduces error due to arithmetic roundoff by addition as well as multiplication while fixed point only introduces error in the addition case. However, the significant advantage of using floating point arithmetic over fixed point is the much broader dynamic range it provides. Both of these effects must therefore be considered when selecting the arithmetic for implementing an algorithm.

The decoupling of the adaptive linearly-constrained filter into fixed-weight and adaptive-weight portions suggests the use of both fixed-point and floating-point in the implementation due to the different features presented in these two weight portions. Fixed-point is preferably used in implementing the fixed-weight portion since roundoff error caused by the computations is minimized and numerical overflow due to the insufficient large of dynamic range can easily be handled via scaling the weights. On the other hand, floating-point is preferably used in implementing the adaptive-weight portion because the weights may be of large values before reaching their optimums. This is true even in a well-posed case that all the optimal weights are small, not to

mention the ill-posed case that some of the optimal weights are large. In addition, the scaling of adaptive weights is far more complicated than that of the fixed weights which makes it infeasible, if not impossible, to scale the weights in the adaptive-weight portion.

When implementing the fixed-weight portion of the canonical form I in fixed-point, noise is introduced by quantizing the coefficients of the matrix $\mathbf{A}$[1]. The study of this quantization noise is very important in practice since quantizing the coefficients in insufficient accuracy can cause substantial violation of the theoretical properties imposed on the matrix $\mathbf{A}$, which may in turn lead to the divergence of the filter. The main purpose of this chapter is to illustrate this quantization noise assuming the matrix $\mathbf{A}$ is constructed via the decomposition procedure outlined in chapter 4. As a result of the quantization of the matrix $\mathbf{A}$ the quantization noise appears both in the reduced-dimensional data vector, $\mathbf{x}_a$, and the quiescent signal, y_q. The quantization noise in $\mathbf{x}_a$ is considered to be more important than that in y_q due to the fact that only the former noise may cause the filter diverges while the latter noise merely lead to a different optimal solution. In addition, the quantization noise in y_q can be minimized by optimizing the filter weights $\mathbf{w}_{qa}$ over the discrete space and/or using longer wordlength in implementing $\mathbf{w}_{qa}$. In order to simplify the problem, only the more complicated quantization noise appearing in $\mathbf{x}_a$ is considered.

As for the adaptive portion, the noise caused by the quantization of the $\mathbf{w}_a$ weights is an extremely difficult problem due to the noise accumulation resulting from the feedback signal. The discussion of this quantization noise has been investigated in [30] and [31] for unconstrained LMS filters. Since the adaptive portion of the adaptive linearly constrained filter is merely an unconstrained filter, the procedures introduced in [30] and [31] can also be applied to analyze the quantization noise caused by $\mathbf{w}_a$. Detailed discussion of this quantization noise, however, is beyond the scope of this thesis.

In Section 5.2, the fixed-point implementation issues including the quantization

[1]Similar quantization noise is introduced by quantizing the coefficients of $\mathbf{B}^{-1}$ in canonical form II. For simplicity, we only discuss canonical form I in this chapter.

procedure, the overflow handling and the number representation systems are discussed. Simulation results in Section 5.3 are used to demonstrate the quantization noise based on both random and deterministic test.

5.2 Fixed-Point Implementation Issues

5.2.1 Successive Quantization Procedure

In finite word length implementations of adaptive linearly constrained filters, the decomposition approach to constructing **A** outlined in Algorithm 1 has an additional significant advantage. This results from the observation that the degradation caused by quantization can be reduced by performing the quantization on the coefficients of the elementary reflector immediately after each iteration. This approach ensures that the required property of **A** is closely preserved at each stage of the iteration.

5.2.2 Overflow Handling

As mentioned in Section 4.5 that the implementation of $\mathbf{A}^{\dagger}\mathbf{x}(n)$ is acheived by connecting a set of PEs in which each PE corresponds to implementing one generic computation. The generic computation is characterized as a 2×2 matrix-vector computation:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{5.1}$$

In order to assure no numerical overflow occurs in the computation, the generic computation is modified by multipling a diagonal matrix **D**:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{DT} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{5.2}$$

where the elements in **D** are used to scale down the two results from the rows of **Tx**. The choice of the scale factors in **D** depends upon the particular composition of **T**. In Section 4.4, various transformations and the corresponding scale factors have been discussed. Of particular interest in this chapter among these transformations are

the Gauss transformation with complex scale factor and the Givens rotation, which represent the non-orthogonal and orthogonal transformations, respectively. These two transformations will be used to demonstrate the quantization noise in Section 5.3. Similarly, this scaling technique can be applied to the generic computations in canonical form II.

5.2.3 Alternative Number Representation Systems

Two fixed-point number systems are considered for use with a specific implementation of the generic computations in the VLSI planar array. These are two's complement (TSC) and power-of-two with at most two nonzero digits (POT2). In both cases, the word size is b bits. With TSC, any or all of these bits may assume values of 0 or 1. In POT2, at most two of the digits (each associated with a sign bit) in the sequence are nonzero. More specifically, a number x is represented by a b-bit TSC as follows:

$$x = \sum_{i=1}^{b} a_i 2^{-i} \tag{5.3}$$

and its complement is

$$\bar{x} = 1 + \sum_{i=1}^{b} \bar{a}_i 2^{-i} \tag{5.4}$$

where

$$a_i \in \{0, 1\}, \bar{1} = 0. \tag{5.5}$$

While a number x is represented by a b-bit POT2 as follows:

$$x = a_1 2^{-b_1} + a_2 2^{-b_2} \tag{5.6}$$

and its complement is

$$\bar{x} = \bar{a}_1 2^{-b_1} + \bar{a}_2 2^{-b_2} \tag{5.7}$$

where

$$a_i \in \{-1, 0, 1\}, \bar{1} = -1, \bar{0} = 0 \tag{5.8}$$

and b_1, b_2 are two integers of values between 1 and b. Since the number of nonzero digits is restricted, the required hardware for a power-of-two multiplier is less than

that of a two's complement multiplier of the same wordlength. The cost of implementing the VLSI planar array is reduced substantially if the coefficients of the generic computations are represented in POT2 rather than in TSC. An efficient VLSI implementation of the POT2 multipler has been proposed in [32]. The quantization effects of implementing $\mathbf{A}$ in both number systems are compared via simulation results in the next section.

5.3 Simulation Results

Since we only consider the quantization noise in $\mathbf{x}_a$ for canonical form I, it is equivalent to considering the quantization noise in $\mathbf{x}_a$ using the GSC structure by assuming that the blocking matrix $\mathbf{W}_s$ is similar to the matrix $\mathbf{A}$. The reason for presenting the quantization material in terms of the GSC structure is that the independent of the $\mathbf{w}_q$ vector on the choice of $\mathbf{W}_s$ matrix simplifies the presentation. Nonetheless, the result presented in this section is directly applicable to canonical form I.

In the simulation, a constraint matrix $\mathbf{C}$ is picked and the corresponding blocking matrix $\mathbf{W}_s$ is constructed using the decomposition procedure. Two ways are used to pick the matrix $\mathbf{C}$ in the simulation: randomly generate $\mathbf{C}$ and deterministically select $\mathbf{C}$. The purpose of the random test is to see the quantization effect based on a statistical point of view and the purpose of the deterministic test is to observe the actual effect in adaptive array design problems. All the constraint matrices used in the simulation are of dimension 32×6. For comparison, both the Gauss transformation (GAT) with complex scale factor and the Givens rotation (GRT) are used to construct the $\mathbf{W}_s$ matrix in the simulation. The fixed-point number systems we consider are TSC and POT2 as defined in Section 5.2.3. The quantization of the $\mathbf{W}_s$ matrix is performed using successive quantization procedure as outlined in Section 5.2.1. Rounding is used in quantizing a number.

Random Generation of C Matrix

Elements of C were chosen to be uniformly and independently distributed between -1 and 1. Once a quantized $\mathbf{W}_s$ matrix had been constructed, the degree to which the quantized matrix remained orthogonal to the C matrix was measured using a performance measure E_1 given by,

$$E_1 = \frac{||\mathbf{W}_s^\dagger \mathbf{C}||_F}{||\mathbf{W}_s||_F ||\mathbf{C}||_F}, \tag{5.9}$$

where $||\cdot||_F$ denotes the Frobenius norm [19]. This measure lies between 0 and 1 and is the normalized error associated with the residue matrix $\mathbf{W}_s^\dagger \mathbf{C}$. The lower bound $E_1 = 0$ is achieved only if $\mathbf{W}_s$ satisfies the orthogonality property and the departure of E_1 from 0 is a quantitative description of the degree to which the space spanned by $\mathbf{W}_s$ overlaps the column space of C. Values of E_1 near 1 indicate that either the column space of C is a subspace of the column space of $\mathbf{W}_s$ or vice versa and therefore not likely to be observed unless extremely coarse quantization is employed.

A total of 32 random constraint matrices were generated. The average value of E_1 obtained by averaging over this set is presented in Figure 5.1 as a function of the word size used in the quantization. Four configurations were investigated, corresponding to the GAT and GRT methods with both TSC and POT2 number systems. The E_1 value for all four cases was about 0.1 for 2-bit quantization and decreased monotonically as the wordlength increased. A saturation wordlength can be defined for each curve as the minimum wordlength beyond which negligible increases in performance are observed. When TSC was used as the number representation system, both GAT and GRT had a saturation wordlength of about 10 bits. In the POT2 case, the GAT had a saturation wordlength of about 6 bits while the GRT had about 4 bits. In addition, the GAT-POT2 had lower overall values for E_1 than did the GRT-POT2. This observation is due to the fact that the superior property of an orthogonal rotation was severely distorted by quantization.

Further, the POT2 approach exhibited a plateau effect for both GAT and GRT such that increasing the number of bits above the threshold point did not result in substantial increased performance. Threshold is observed due to the fact that the

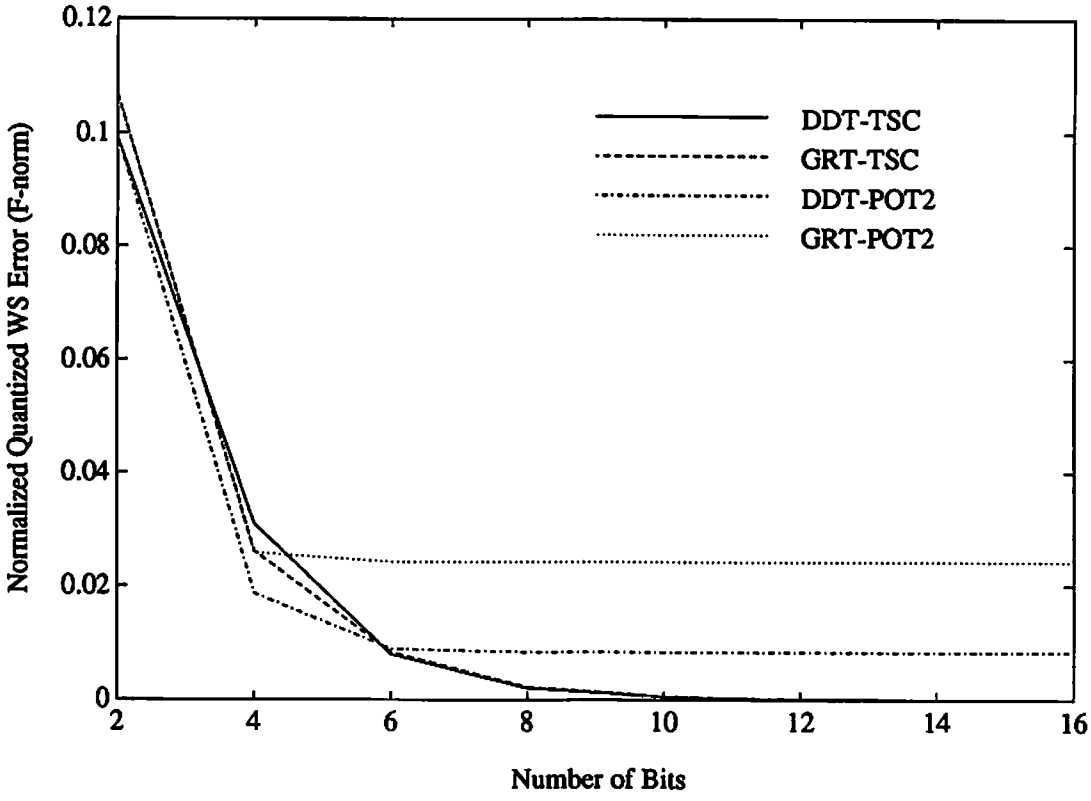

Figure 5.1: Normalized quantization error for the random C matrix example.

number of nonzero bits in POT2 representation was restricted to two and an upper limit on performance must exist, regardless of the wordlength. Improvements can be obtained by increasing the number of nonzero bits but only at the expense of higher implementation cost.

Comparison of the results in Fig. 5.1 reveals that the quantization effect observed with GAT and GRT was about the same when $\mathbf{W}_s$ was implemented using a TSC number system. However, this was not the case when POT2 representation was employed as performance of GAT was uniformly better than that of GRT, regardless of wordlength. The difference was greater at longer wordlengths. Specifically, the measured value of E_1 for the GAT-POT2 was about 0.0088 at the saturation wordlength and about 0.0260 for the GRT-POT2 method. This latter value indicates that the $\mathbf{W}_s$ with orthogonal columns (GRT) was more sensitive to the quantization than that with non-orthogonal column (GAT).

Deterministic C Matrix Experiment

A linearly-distributed 32 element adaptive narrowband antenna array with six constraints was selected as the example for the deterministic tests. The element spacing was one-half wavelength and the six constraints consisted of five gain response point constraints and one additional constraint to control the quiescent response of the array [33]. The gain control constraints provided unity gain in the broadside direction and nulls in four directions: $12^0, 14^0, 16^0$ and 18^0. Figure 5.2 illustrates the infinite precision optimal gain response pattern for this example. The input signal environment for the array consisted of a 0 dB desired signal incident from the broadside $\theta_S = 0^0$ direction, a single directional interference (jammer) incident from $\theta_J = 22^0$ at a power level of 20 dB, and white, uncorrelated noise at a 0dB power level. The covariance matrix for this data set is given by,

$$\mathbf{R}_{xx} = \mathbf{I} + \mathbf{v}(\theta_S)\mathbf{v}^\dagger(\theta_S) + 100 \cdot \mathbf{v}(\theta_J)\mathbf{v}^\dagger(\theta_J) \ , \tag{5.10}$$

where $\mathbf{v}(\theta)$ is the array steering vector for direction of arrival θ.

Performance curves for this example were obtained by comparing the suboptimal array weights $\hat{\mathbf{w}}_{opt}$ obtained using a specific quantizer rule with the unquantized

optimal weight vector $\mathbf{w}_{opt}$. The latter was computed from Eq. (2.17) using double-precision floating-point calculations on a VAX-11 computer. In computing the quantized suboptimal responses, only the effects of quantization in $\mathbf{W}_s$ were considered. Thus, quantized $\mathbf{W}_s$ matrices were computed according to the particular rule under consideration (GAT-POT2, GRT-TSC, etc.) and the approximation to the optimal response $\hat{\mathbf{w}}_{opt}$ was calculated with double-precision floating point using Eq. (2.17).

In order to evaluate the performance obtained under the four methods for constructing and implementing $\mathbf{W}_s$, two measures were employed. The first evaluated the norm of the vector difference between $\mathbf{w}_{opt}$ and $\hat{\mathbf{w}}_{opt}$,

$$E_2 = \frac{||\hat{\mathbf{w}}_{opt} - \mathbf{w}_{opt}||_2}{||\mathbf{w}_{opt}||_2} . \tag{5.11}$$

The second indicated the degree to which the constraint equation was satisfied,

$$E_3 = \frac{||\hat{\mathbf{f}} - \mathbf{f}||_2}{||\mathbf{f}||_2} , \tag{5.12}$$

where,

$$\hat{\mathbf{f}} = \mathbf{C}^{\dagger}\hat{\mathbf{w}}_{opt}. \tag{5.13}$$

Note that both measures are zero if $\mathbf{W}_s$ satisfies the orthogonality property. The measures E_2 and E_3 for the above design example are shown in Figure 5.3 as a function of processor wordlength.

The distortion introduced by the quantization of the $\mathbf{W}_s$ was also measured by plotting the magnitude of array response. The complex array response $b(\theta)$ at an incident angle θ is given as,

$$b(\theta) = \mathbf{v}^{\dagger}(\theta)\mathbf{w}_{opt} . \tag{5.14}$$

Plots of $|b(\theta)|$ are shown in Figure 5.4 for the two's complement quantized processors, GAT-TSC and GRT-TSC computed at 8-bit precision. Comparable results for the power-of-two cases, GAT-POT2 and GRT-POT2, are illustrated in Fig. 5.5. Comparison of these results with Fig. 5.2 shows that the performance measures E_2 and E_3 reflect the actual array response performance in the sense that the solutions which have low values of the performance measure also have array responses which agree most closely with the unquantized case.

Discussion

The results presented in Fig. 5.3 show dramatically improved performance for the 2-bit word size case. At least two reasons can be cited for this seemingly anomalous behavior. First, the process described in this paper for generating a quantized approximation to the $\mathbf{W}_s$ matrix consists of a stage-by-stage procedure. This approach will not, in general guarantee that a global optimal solution has been achieved. The latter can only be found by searching the entire set of discrete approximations for those points which produce global minima and this represents a truly formidable numerical task even for moderate-sized problems. It should be noted, however, that quantization by stage described in this paper *does* result in a local minimum at the current stage. It does seem unlikely, however, that this characteristic fully explains the performance levels observed for the 2-bit word size examples.

The second, and more meaningful, reason for the observed behavior is related to the particular geometric properties of the example selected. In linearly-constrained minimization problems, it is convenient to describe weight vectors for the processor in terms of their projections onto the constraint subspace and onto the orthogonal complement of this space. Any vector which satisfies the constraints must have a predefined projection onto the constraint subspace. Different vectors, which all satisfy the given constraints can therefore only differ in the degree to which they project onto the orthogonal complement. For the particular example selected above, the optimal vector $\mathbf{w}_{opt}$ was contained largely in the space spanned by the constraint vectors. Quantitatively, the l2-norm of the projection in this subspace was 0.1962 while that in the orthogonal complement was only 0.0075. Simply quantizing this value to zero results in an E2 value of 0.9632 which is lower that that observed for both 4-bit and 6-bit wordsizes in GAT-POT2 and GRT-POT2. Using few quantization levels, such as in the 2-bit case, increases the likelihood that the orthogonal complement value will be set to zero.

A second example was generated in which a greater portion of the optimal vector resided in the orthogonal complement. In this case, the same 32 element array was

used, but the signal, jammer and constraint conditions were changed. The new parameters were: Unity gain in the desired signal direction $\theta = 0^0$; four null constraints in the directions 22^0, 24^0, 26^0 28^0; and a jammer direction of 32^0. The relative powers of the signal, jammer and white noise were unchanged. Under these conditions, the projections of $\mathbf{w}_{opt}$ in the constraint and orthogonal complement spaces have norms of 0.2664 and 0.0567, respectively. Figure 5.3 illustrates the behavior of the measures E_2 and E_3 for this example. Although the results are not strictly monotonic, the overall behavior does indicate the expected improvement in performance with increased wordsize.

Figures 5.3 and 5.6 illustrate that the GAT-TSC, GRT-TSC and GAT-POT2 can provide acceptable performance for sufficient long wordlength but that GRT-POT2 cannot. This conclusion agrees with that of the random test.

5.4 Summary

This chapter has discussed the issues of overflow handling and alternative number representation associated with implementing the fixed-weight portion of the adaptive linearly-constrained filter in fixed-point arithmetic. A simple number system have been proposed and shown to result in significant hardware complexity reduction for a VLSI implementation. Finally, the effects of finite wordlength implementation have been investigated using simulation results. These preliminary findings are encouraging and suggest that relatively small wordlengths can provide effective processing within the fixed-weight portion of the adaptive linearly-constrained filter.

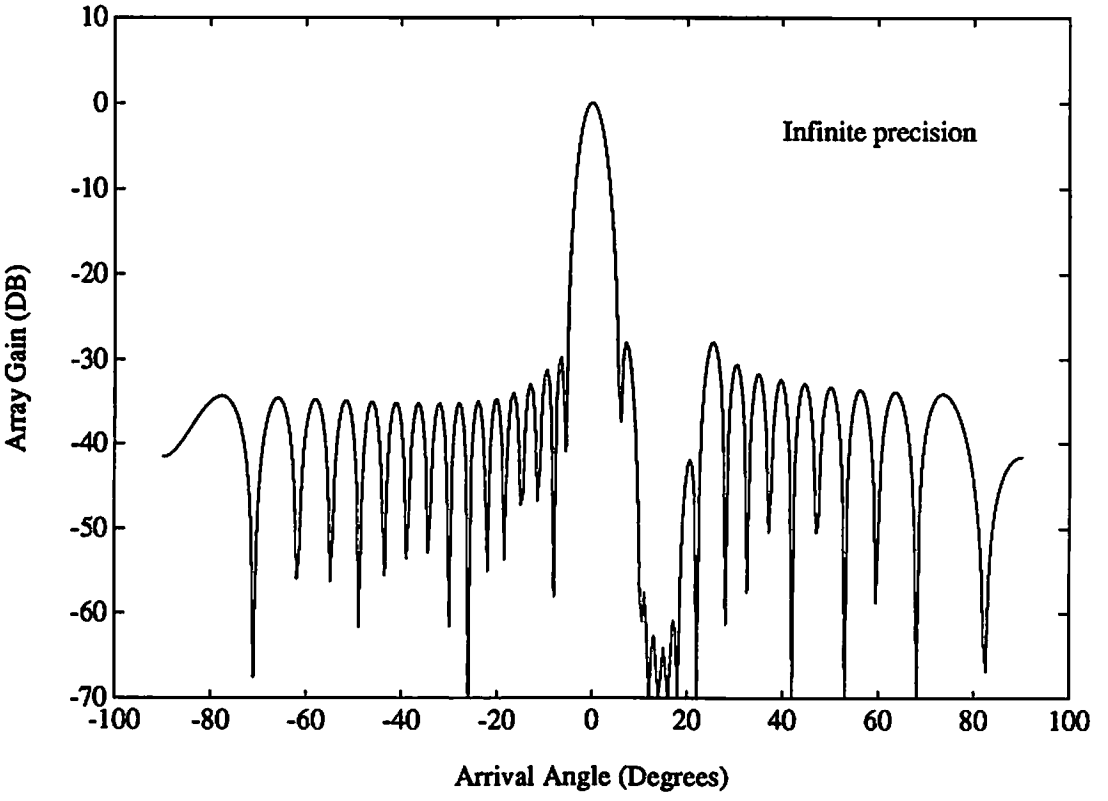

Figure 5.2: Optimal array response for the deterministic C matrix example.

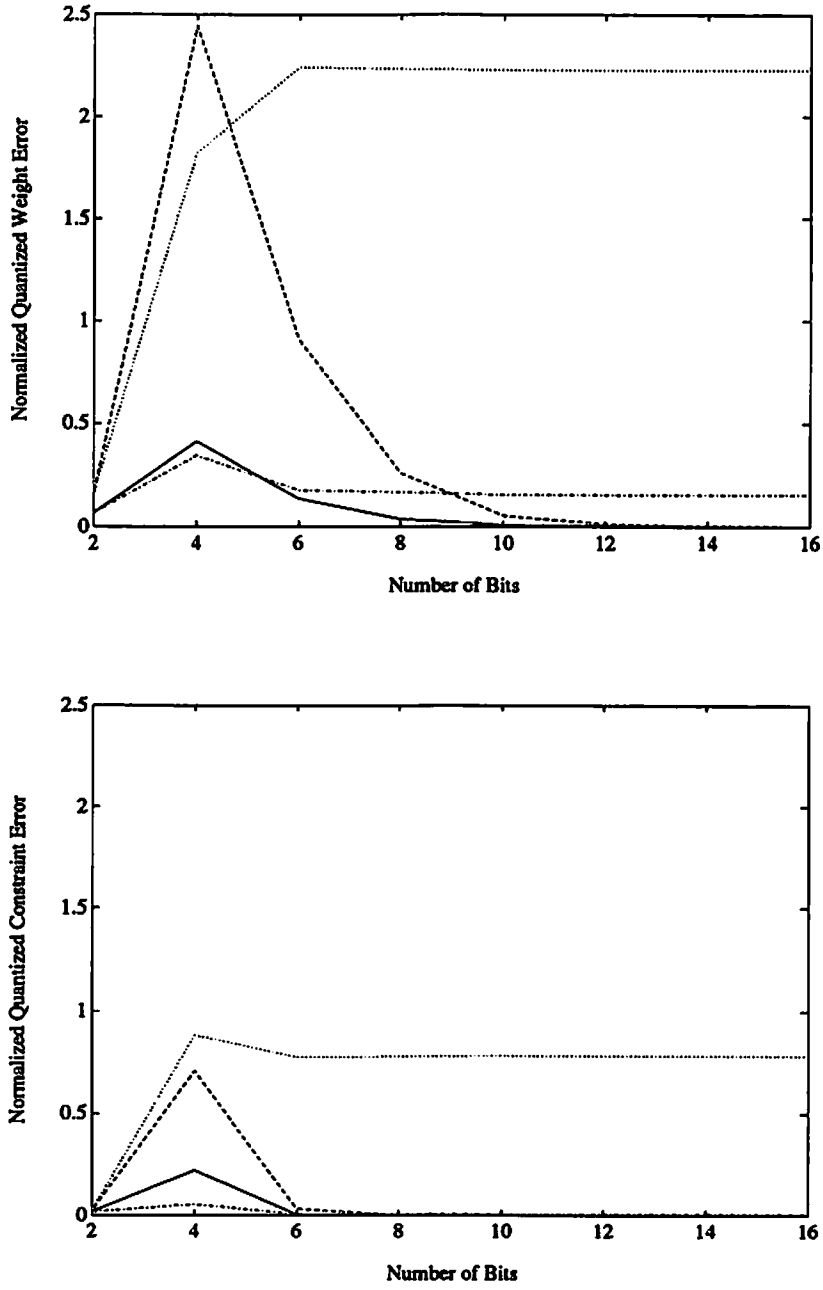

Figure 5.3: Normalized quantization error for the adaptive array example.

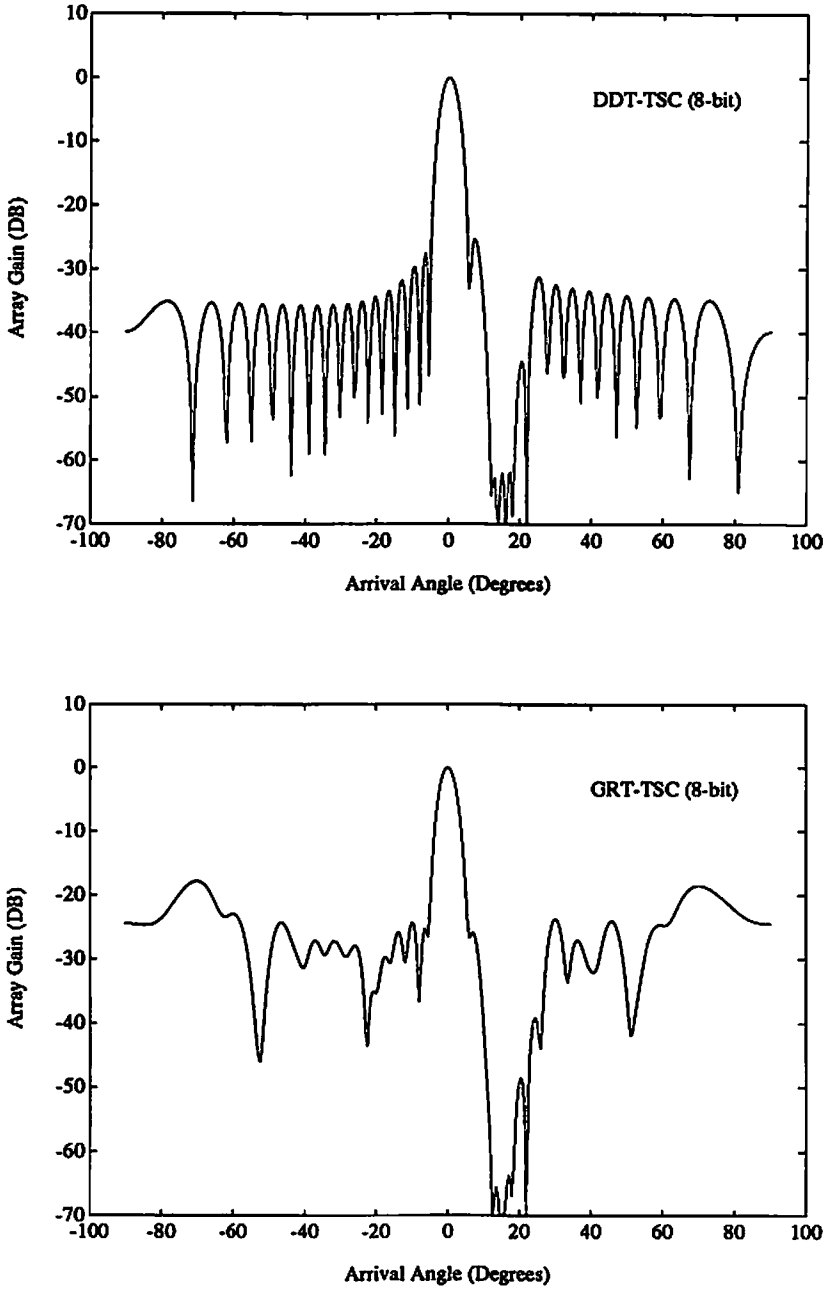

Figure 5.4: The magnitude array responses in two's complement number system.

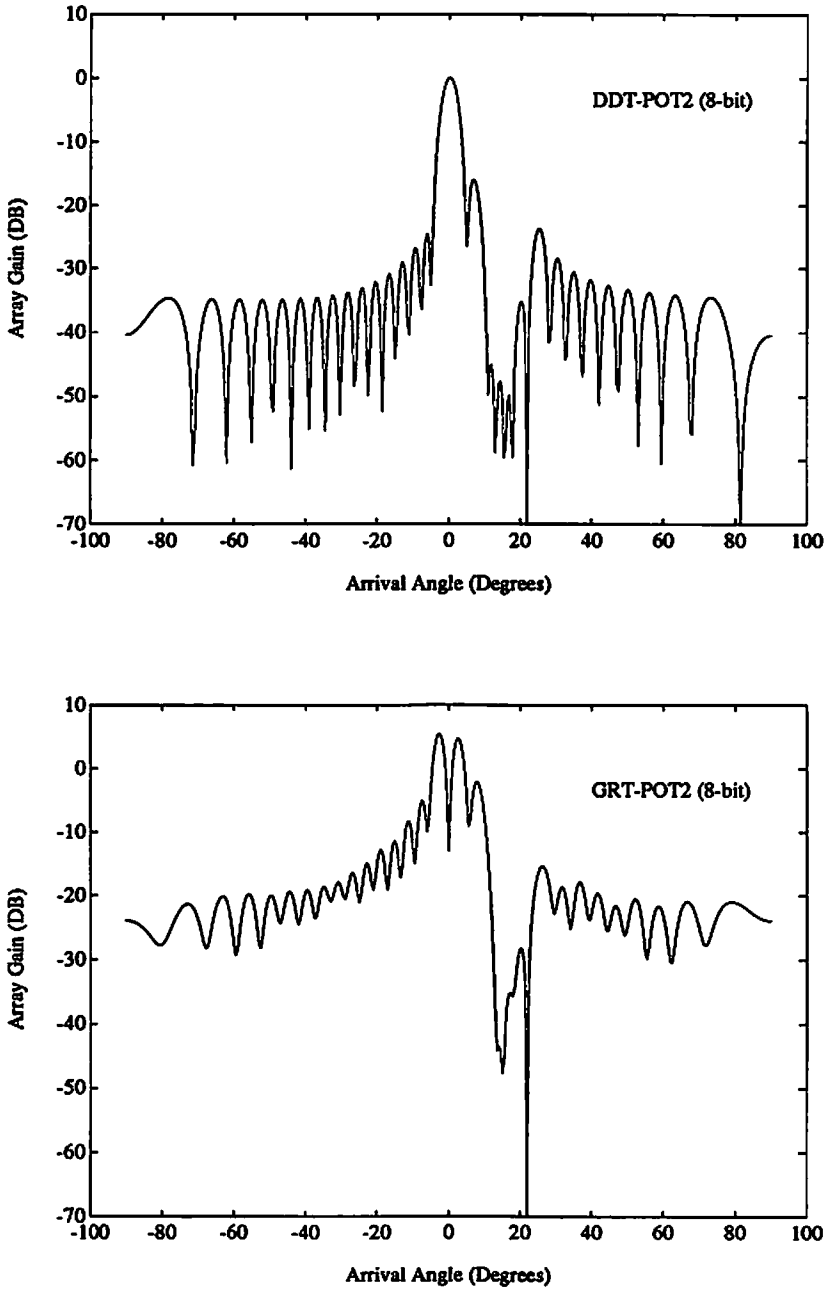

Figure 5.5: The magnitude array responses in power-of-two number system.

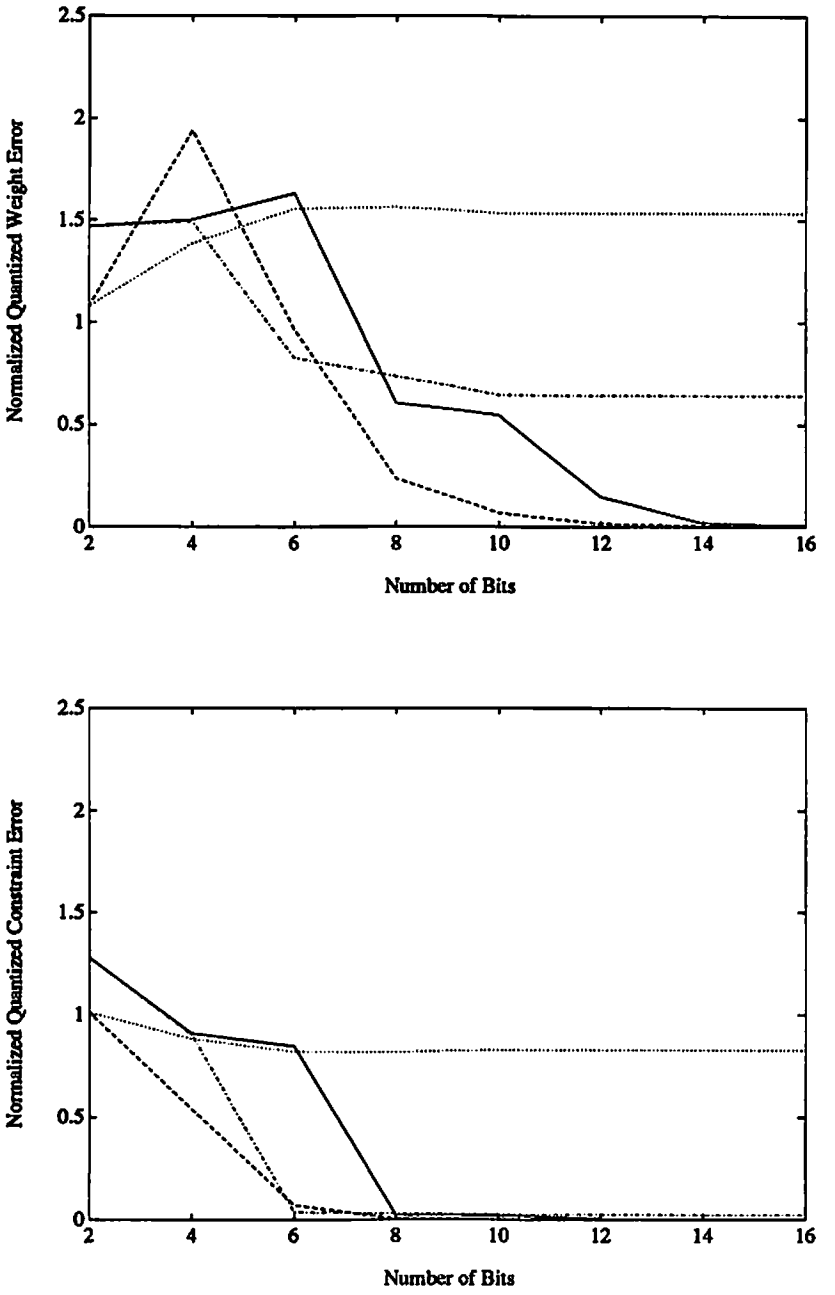

Figure 5.6: Normalized quantization error for the second adaptive array example.

Chapter 6

A Generalization to Linearly-Constrained Filters

6.1 Introduction

This chapter addresses the problem of linearly-constrained filtering in a more general case in which the linear constraints in the filter are not fixed. Instead, the righthand sides of the constraint equations are allowed to be adjustable. The filter of this type is termed as having *adjustable* constraints. The application of this generalized linearly-constrained filter requires further investigation and is beyond the scope of this thesis. The main effort in this chapter therefore lies only in deriving an efficient implementation structure.

To formulate adjustable linear constraints, we incorporate a time variable n in the response vector $\mathbf{f}$ in the constraint specification given in previous chapters,

$$\mathbf{C}^{\dagger}\mathbf{w} = \mathbf{f}(n). \tag{6.1}$$

To implement an adaptive filter with adjustable linear constraints, a structure that

immediately follows from the MGSC is to allow $\mathbf{w}_{qm}$ to be adjustable according to the change of $\mathbf{f}$. However, the difficulty with this structure is that the weights $\mathbf{w}_a(n)$ will not start converging until the vector $\mathbf{w}_{qm}$ is held fixed since the update of $\mathbf{w}_a(n)$ depends on $y_q(n)$. In the case of changing $\mathbf{f}$ at every time instance the filter will not converge at all since it has to keep tracing different desired signals at different time instance. Even in the case of infrequently changing $\mathbf{f}$ such that there is sufficient time for $\mathbf{w}_a(n)$ to converge, this structure still bears a disadvantage of the need of re-adjusting $\mathbf{w}_a(n)$ every time $\mathbf{f}$ changes.

To alleviate these problems, a second structure as shown in Figure 6.1 is considered. This new structure mainly consists of two portions: The first is a transformation, denoted by $\mathbf{W}_c(n)$, which maps the input vector, $\mathbf{x}(n)$, into a M-dimensional vector, $\mathbf{z}(n)$, and the second is a vector, denoted by $\mathbf{g}(n)$, whose elements linearly combine with $\mathbf{z}(n)$ to produce the final output, $y(n)$. The significant features of this structure are that the transformation $\mathbf{W}_c(n)$ can be updated independent of $\mathbf{f}(n)$ and the vector $\mathbf{g}(n)$ is a simple linear mapping from $\mathbf{f}(n)$. Consequently, the convergence of the weight matrix $\mathbf{W}_c(n)$ is not affected by the values of the elements in $\mathbf{f}(n)$. Once $\mathbf{W}_c(n)$ converges, the reconfiguration of the filter weights due to the change of $\mathbf{f}(n)$ simply involves a relatively simple linear mapping to obtain $\mathbf{g}(n)$.

In Section 6.2, it is shown that the optimal weights of a linearly-constrained filter can always be expressed as a linear combination of a set of primitive weights. Based on this linear combination expression and the MGSC algorithm, the detailed implementation of the structure in Figure 6.1 is derived in Section 6.3. The chapter ends with several concluding remarks on the generalization of the linearly-constrained filter with adjustable constraints.

6.2 Mathematical Groundwork

This section shows that the optimal weight vector of a linearly-constrained filter with M constraints can always be expressed as a linear combination of M components. In the first part of this section, these M components are first identified as the optimal

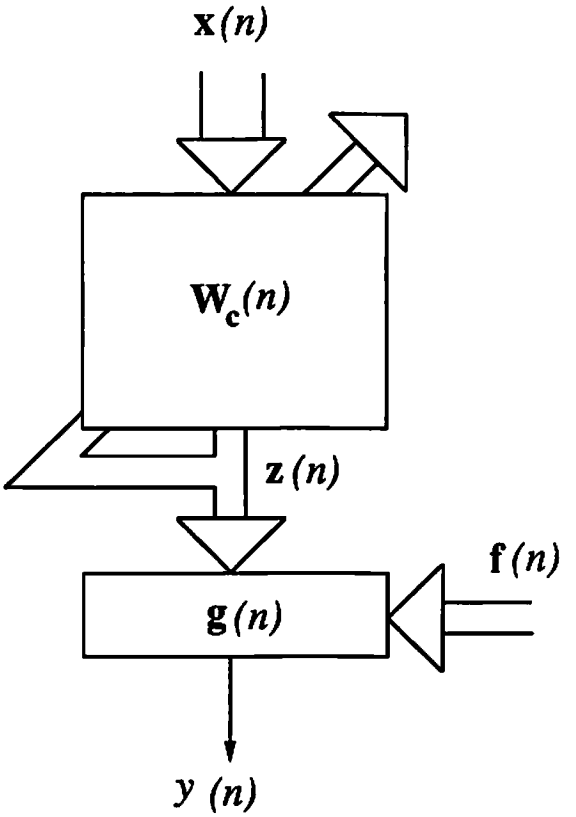

Figure 6.1: The implementation structure of the adaptive filter with adjustable constraints.

solutions to a class of standard linearly-constrained problem with constraints induced by the original constraint matrix; in the second part, it is shown that the overall optimal weight vector can be obtained by linearly combining these M components.

6.2.1 Standard Linearly Constrained Problems

Given a constraint matrix C of dimension $N \times M$, we can define M generic sets of linear constraint equations as,

$$\mathbf{C}_i^\dagger \mathbf{w} = \mathbf{e}_i \quad \text{for} \quad i = 1, 2, \cdots, M \ , \tag{6.2}$$

where $\mathbf{C}_i$ is a submatrix of C containing the first i columns and $\mathbf{e}_i$ is a $i \times 1$ vector with unity as its only nonzero component appearing at the last entry. Starting with index i from 1 to M, the i_th set of the linear constraints restricts that the linear combination of the weight vector with the i_th column of C is equal to unity while that with the previous columns are zero.

A class of standard linearly-constrained minimization problem induced by C can now be defined as minimizing the filter output power subject to the M generic sets of linear constraints. Accordingly, there are M optimal weight vectors, denoted by $\mathbf{w}_{c1}, \mathbf{w}_{c2}, \cdots, \mathbf{w}_{cM}$, associated with this class of linearly-constrained minimization problem. It is shown in Appendix A that each $\mathbf{w}_{ci}$ is given by

$$\mathbf{w}_{ci} = (\mathbf{H}^\dagger)^{-1} \mathbf{P}_i \mathbf{H}^\dagger \mathbf{c}_i \tag{6.3}$$

where $\mathbf{H}$ is the square-root of $\mathbf{R}_{xx}$ and $\mathbf{P}_i$ is the projection matrix of $\mathbf{H}^{-1}\mathbf{C}_i$. These optimal weight vectors are referred to here as *primitive weight vectors.*

6.2.2 Generating optimal weights from primitive weights

The following theorem shows that the optimal weight vector of the linearly-constrained filter can be obtained by properly combining the corresponding primitive weight vectors. The combination coefficients, however, depends on the response vector f.

Theorem 5 *Consider the minimization problem of*

$$\min_{\mathbf{w}} \mathbf{w}^{\dagger}\mathbf{R}_{xx}\mathbf{w} \tag{6.4}$$

subject to the linear constraints

$$\mathbf{C}^{\dagger}\mathbf{w} = \mathbf{f} \tag{6.5}$$

where

$$\mathbf{C} = [\mathbf{c}_1 \ \mathbf{c}_2 \ \cdots \ \mathbf{c}_M] \tag{6.6}$$

and

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_M \end{bmatrix} \tag{6.7}$$

are the constraint matrix and response vector, respectively. With the assumption that $\mathbf{c}_i$*'s are mutually orthogonal*[1]*, the optimal solution for this minimization problem is a linear combination of the primitive weight vectors* $\mathbf{w}_{ci}$*'s with appropriate coefficients* g_i*'s, i.e.,*

$$\mathbf{w}_{opt} = \sum_{i=1}^{M} g_i \mathbf{w}_{ci}. \tag{6.8}$$

The primitive weight $\mathbf{w}_{ci}$ *is given by Eq.(6.3), and the vector* $\mathbf{g}$ *consisting of the combination coefficients is given by*

$$\mathbf{g} = \mathbf{T}^{-1}\mathbf{f} \tag{6.9}$$

where

$$\mathbf{T} = \mathbf{C}^{\dagger}\mathbf{W}_c \tag{6.10}$$

and

$$\mathbf{W}_c = [\mathbf{w}_{c1} \ \mathbf{w}_{c2} \ \cdots \ \mathbf{w}_{cM}]. \tag{6.11}$$

In addition, $\mathbf{T}$ *is a triangular matrix with unities on the diagonal.*

[1]This assumption can be asserted without loss of generality since an arbitrary given constraint specification can always be reformulated such that the corresponding constraint matrix has orthogonal columns.

Proof:

By starting with only one constraint and successively adding one constraint at a time in the filter, the optimal weight vector at the $(i+1)\_th$ stage, $\mathbf{w}_{i+1}$, is related to that at the i_th stage, $\mathbf{w}_i$, by the following recursion,

$$\mathbf{w}_{i+1} = \mathbf{w}_i + (f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i)\mathbf{w}_{ci+1}. \tag{6.12}$$

(The proof of this recursion can be found in Appendix A.) Recursively substituting this recursion from index i equals 1 to M, it is easy to show that the optimal weight vector with M constraints is a linear combination of the primitive weight vectors $\mathbf{w}_{ci}$, for $i = 1, 2, \cdots, M$, i.e.,

$$\mathbf{w}_{opt} = \sum_{i=1}^{M} g_i \mathbf{w}_{ci} \tag{6.13}$$

or in a vector form

$$\mathbf{w}_{opt} = \mathbf{W}_c \mathbf{g}. \tag{6.14}$$

By directly substituting the above expression of $\mathbf{w}_{opt}$ into the constraint equations

$$\mathbf{C}^{\dagger}\mathbf{w}_{opt} = \mathbf{f}, \tag{6.15}$$

it follows that

$$\mathbf{T}\mathbf{g} = \mathbf{f} \tag{6.16}$$

where

$$\mathbf{T} = \mathbf{C}^{\dagger}\mathbf{W}_c. \tag{6.17}$$

To prove that $\mathbf{T}$ is a triangular matrix, we can check the i_th column of $\mathbf{T}$ given by

$$\mathbf{C}^{\dagger}\mathbf{w}_{ci}. \tag{6.18}$$

Since $\mathbf{w}_{ci}$ is the i_th primitive weight vector, it must satisfy the constraint condition

$$\mathbf{C}_i^{\dagger}\mathbf{w}_{ci} = \mathbf{e}_i. \tag{6.19}$$

This implies that the first i elements of $\mathbf{C}^{\dagger}\mathbf{w}_{ci}$ is equal to $\mathbf{e}_i$ and $\mathbf{T}$ is therefore a lower triangular matrix with unities on the diagonal.

Q.E.D.

Based on the above theorem, the computation of the filter output

$$y(n) = \mathbf{w}_{opt}^{\dagger}\mathbf{x}(n) \tag{6.20}$$

can be obtained by first computing an intermediate output

$$\mathbf{z}(n) = \mathbf{W}_c^{\dagger}\mathbf{x}(n) \tag{6.21}$$

and then combining this output with $\mathbf{g}$,

$$y(n) = \mathbf{g}^{\dagger}\mathbf{z}(n). \tag{6.22}$$

In the next section, a structure for implementing the adaptive filter with adjustable linear constraints is derived based on the use of this two-step computation and the MGSC algorithm.

6.3 An Adaptive Implementation Structure

The two-step computation mentioned in last section naturally divides the implementation of the adaptive filter with adjustable constraints into two parts corresponding to implementing $\mathbf{W}_c$ and $\mathbf{g}$. Adaptive implementation of the first part is equivalent to adaptively implementing the class of standard linearly-constrained minimization problem as defined in last section. It is therefore possible to implement it by M individual MGSC structures where each MGSC implements one minimization problem. However, this MGSC implementation can be further simplified by using the fact that the righthand sides of the constraint equations are particularly simple and the constraint matrices in the minimization problems are formed recursively by adding one constraint vector at a time. On the other hand, as a direct result of the above theorem, implementing $\mathbf{g}$ according to the change of $\mathbf{f}$ simply involves forming $\mathbf{T}$ from $\mathbf{W}_c$ and solving a triangular linear system. In the following, the adaptive implementation of the weight matrix $\mathbf{W}_c$ and the reconfiguration of the combination coefficients $\mathbf{g}$ are given in detail.

6.3.1 Adaptive implementation of the weight matrix $\mathbf{W}_c$

The vector $\mathbf{z}(n)$ is computed from the input vector $\mathbf{x}(n)$ as follows:

$$\mathbf{z}(n) = \mathbf{W}_c^{\dagger}(n)\mathbf{x}(n). \tag{6.23}$$

In order to apply the MGSC algorithm, we have to implement this computation in a transformed domain specified by a nonsingular matrix $\mathbf{A}$ which transforms the constraint matrix $\mathbf{C}$ into $\hat{\mathbf{C}}$ of canonical form as given by Eq.(3.25). It was proved in Chapter 3 that by using this $\mathbf{A}$ matrix the original minimization problem can be equivalently expressed in the transformed domain as

$$\min_{\hat{\mathbf{w}}} \hat{\mathbf{w}}^{\dagger}\hat{\mathbf{R}}_{xx}\hat{\mathbf{w}} \tag{6.24}$$

subject to the linear constraints

$$\hat{\mathbf{C}}^{\dagger}\hat{\mathbf{w}} = \mathbf{f}(n) \tag{6.25}$$

where $\hat{\mathbf{R}}_{xx}$ is the covariance matrix of the transformed signal $\hat{\mathbf{x}}(n)$ as defined in Eq.(3.31).

In general, the columns of $\hat{\mathbf{C}}$ are not mutually orthogonal. This prevents directly applying Theorem 5 to implement $\mathbf{z}(n)$ in the transformed domain. However, noting that by multiplying both sides of Eq.(6.25) by $(\mathbf{V}^{\dagger})^{-1}$ we have

$$\tilde{\mathbf{C}}^{\dagger}\hat{\mathbf{w}} = (\mathbf{V}^{\dagger})^{-1}\mathbf{f}(n) \tag{6.26}$$

where

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{6.27}$$

Accordingly, the constraint matrix $\tilde{\mathbf{C}}$ now has orthogonal columns and the result of Theorem 5 can be applied.

Let the primitive weights in the transformed domain with the constraint matrix $\tilde{\mathbf{C}}$ be denoted by $\hat{\mathbf{W}}_c$. The computation of $\mathbf{z}(n)$ can now be preformed in the transformed domain as

$$\mathbf{z}(n) = \hat{\mathbf{W}}_c^{\dagger}(n)\hat{\mathbf{x}}(n) \tag{6.28}$$

where the time variable n is included in $\hat{\mathbf{W}}_c$ since the primitive weights are implemented adaptively. To simply the derivation, the vector $\hat{\mathbf{x}}(n)$ and the matrix $\hat{\mathbf{W}}_c(n)$ are partitioned as the upper $N-M$ and the lower M components:

$$\hat{\mathbf{x}}(n) = \begin{bmatrix} \mathbf{x}_a(n) \\ --- \\ \mathbf{x}_q(n) \end{bmatrix} \tag{6.29}$$

and

$$\hat{\mathbf{W}}_c(n) = \begin{bmatrix} -\mathbf{W}_a(n) \\ --- \\ \mathbf{T}(n) \end{bmatrix}. \tag{6.30}$$

From Theorem 5, $\mathbf{T}(n)$ is a lower triangular matrix with unities on the diagonal:

$$\mathbf{T}(n) = \begin{bmatrix} 1 & 0 & \cdot & 0 & 0 \\ -t_{21}(n) & 1 & 0 & \cdot & 0 \\ -t_{31}(n) & -t_{32}(n) & 1 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ -t_{M1}(n) & -t_{M2}(n) & \cdot & -t_{M,M-1}(n) & 1 \end{bmatrix} \tag{6.31}$$

The *i_th* component of $\mathbf{z}(n)$ is then equal to

$$z_i(n) = \mathbf{t}_i^{\dagger}(n)\mathbf{x}_q(n) - \mathbf{w}_{ai}^{\dagger}(n)\mathbf{x}_a(n) \tag{6.32}$$

where $\mathbf{t}_i(n)$ and $\mathbf{w}_{ai}(n)$ are the *i_th* columns of $\mathbf{T}(n)$ and $\mathbf{W}_a(n)$, respectively. The adaptive weights in these two vectors are updated such that the variance of $z_i(n)$ is minimized.

The block diagram for adaptively implementing $\mathbf{z}(n)$ is shown in Figure 6.2. The input vector $\mathbf{x}(n)$ is first transformed into $\mathbf{x}_a(n)$ and $\mathbf{x}_q(n)$ by the matrix $\mathbf{A}$; $\mathbf{x}_a(n)$ and $\mathbf{x}_q(n)$ are then multiplied by the Hermitian transpose of $\mathbf{W}_a(n)$ and $\mathbf{T}(n)$, respectively, to generate $\mathbf{y}_q(n)$ and $\mathbf{y}_a(n)$; $\mathbf{z}(n)$ is obtained by subtracting $\mathbf{y}_a(n)$ from $\mathbf{y}_q(n)$ and is used to update the weights specified in $\mathbf{W}_a(n)$ and $\mathbf{T}(n)$; the weight update is done such that the weights in the *i_th* columns of $\mathbf{W}_a(n)$ and $\mathbf{T}(n)$ is adjusted to minimize the *i_th* components of $\mathbf{z}(n)$. The computation involved in this structure is of order $O(NM)$ provided that the LMS algorithm is used for weight update and the matrix $\mathbf{A}$ is of decomposed form as described in Chapter 4.

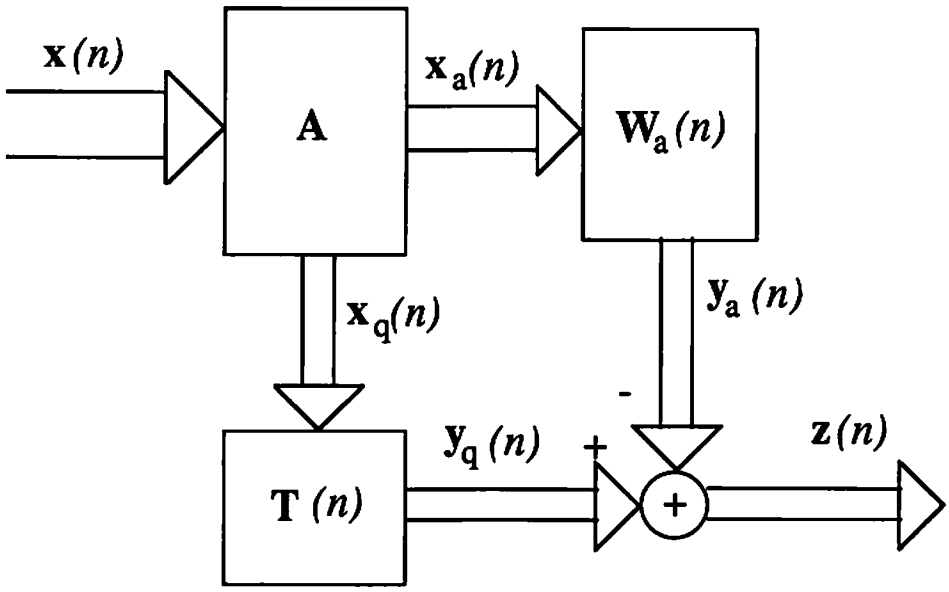

Figure 6.2: The detailed implementation of the weight matrix $\mathbf{W}_c(n)$.

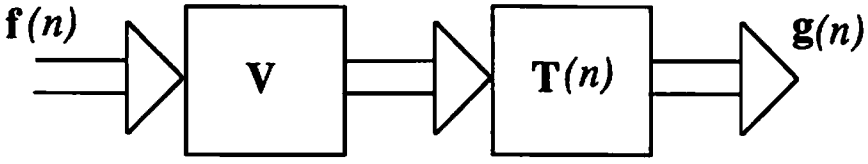

Figure 6.3: The reconfiguration of the response vector $\mathbf{f}(n)$.

6.3.2 Reconfiguration of $\mathbf{g}(n)$

According to Theorem 5, the combination coefficients contained in $\mathbf{g}(n)$ must satisfy the following equality,

$$\mathbf{T}(n)\mathbf{g}(n) = (\mathbf{V}^{\dagger})^{-1}\mathbf{f}(n). \tag{6.33}$$

Reconfigurating $\mathbf{g}(n)$ by the change of $\mathbf{f}(n)$ thus simply involves solving two $M \times M$ triangular linear systems as represented in Figure 6.3. The final filter output, $y(n)$, is obtained by the inner product of $\mathbf{g}(n)$ and $\mathbf{z}(n)$.

6.4 Summary

This chapter has presented a new structure for implementing the adaptive filter with adjustable constraints. The implementation is based on first using a class of standard linearly-constrained filters with constraint matrices induced from the original constraint matrix and then combining the output of these filters to generate the final output. The significant feature of this two step implementation is that the feedback signals used to adjust the filter weights do not depend on the particular response vector. It is therefore possible to change the response vector at any moment without affecting the convergence of the weights. Implementing the first step is based on the use of the MGSC algorithm while implementing the second step simply involves two triangular linear system solvers and one linear combiner. The resulting structure was referred to as the Extended Generalized Sidelobe Canceller (EGSC). The application of the EGSC requires further investigation and is beyond the scope of this thesis.

Chapter 7

Summary and Future Research Directions

7.1 Summary

This thesis has presented an extensive study of the adaptive linearly-constrained filter. The design of an adaptive linearly-constrained filter for a particular application can be typically divided into two major steps, one is to set up the linear constraints and the other is to select an implementation structure. The material presented in this thesis is selected with the effort of providing some guidelines for these two design steps. The analysis of the steady-state performance in Chapter 2 helps evaluate the performance of the constraints; the comparison of three iterative algorithms in Chapter 3 unifies the implementation algorithms; the derivation of the dual implementation structure in Chapter 4 ensures the implementation efficiency for arbitrary number of constraints; the study of the quantization effects in Chapter 5 suggests a trade-off between the implementation cost and system performance; and, finally, the generalization of the adaptive linearly-constrained filter in Chapter 6 indicates a future extension in adaptive filters.

7.2 Future Research Directions

In the following, some possible directions for future research are summarized.

1) Constraint Specification: The methods of designing linear constraints for adaptive filters have been widely studied, especially in adaptive array areas. Essential considerations of selecting a set of constraints are the effectiveness of the constraints and the number of constraints. Previous works on the constraint specification include the use of point, derivative and eigenvector constraints [13]. There has also been some work [34] on applying additional constraints to the adaptive arrays in order to reduce the dimension of adaptive weights.

The study of the linearly-constrained filter in Chapter 2 suggests some possible directions for future research in constraint specification. For instance, a set of constraints can be designed to maximize the optimal SNR given in Section 2.5 by using a pre-specified input covariance matrix. This covariance matrix may be formed by using some *a priori* knowledge about the desired and jammer signals. Moreover, the recursive formulas of the optimal weight and output power expression derived in Section 2.6 provide some useful insights to evaluate the effect of adding one additional constraint to the filter. This in turn helps determine the number of constraints to be utilized in the filter.

2) Convergence Analysis: As shown in Chapter 3, an adaptive linearly constrained filter can always be factorized as a cascaded of fixed and adaptive weights portions. In Chapter 4, we have focused on deriving effective structures to reduce the required computational complexity in the implementation of the fixed-weight portion. However, since the fixed-weight implementation is not unique the convergence properties of the adaptive weight portion may be different for various fixed-weight implementation structures. The analysis of these properties requires further investigation.

3) Quantization Studies: In Chapter 5, the quantization effects on the fixed-weight implementation has been studied via some preliminary experimental results. The simulations have shown that it is possible to implement the fixed-weight portion with relatively low precisions. To provide more general study on this issue in the future, we suggest to establish a quantization noise analysis model for the fixed-weight portion

implemented in decomposed forms.

4) Extensions of the Linearly-Constrained Filter: A structure for implementing an adaptive filter with adjustable constraints has been derived in Chapter 6. One application of this structure which has been suggested in [35] is to overcome the sensitivity problems encountered in linearly-constrained adaptive arrays. More research is required in order to apply this structure to many other areas in signal processing.

Appendix A

The main purpose of this appendix is to derive a recursive update formula for the optimal weight vector of a linearly-constrained filter in which the update is caused by adding one additional constraint to the filter. We would like to point out here that this recursive formula is only true under the orthogonal constraint matrix assumption that the constraint vectors are mutually orthogonal and of unity length. Without this orthogonality assumption, the recursive formula will appear in a more complicated form. To simply the problem, we only consider tha case when this assumption is valid. However, this does not limit the generality of the results in this appendix since for *any* given linear constraint equations it is always possible to reformulate them into a specification which possesses the orthogonality assumption.

This appendix is organized as follows. Some notations and definitions used to derive the recursive update formula are first given. Important matrix identities are then addressed in five lemmas. Finally, a main theorem is proved using these matrix identities.

Notations and Definitions

Let the constraint equations of a linearly-constrained filter with i constraints be given by

$$\mathbf{C}_i^{\dagger}\mathbf{w} = \mathbf{f}_i. \qquad (\mathrm{A}-1)$$

The matrix $\mathbf{C}_i$ is assumed full rank and of size $N \times i$. Adding one more constraint to this filter corresponds to appending one column vector $\mathbf{c}_{i+1}$ to $\mathbf{C}_i$ and one element

f_{i+1} to $\mathbf{f}_i$. Accordingly, the new constraint equations are given by

$$\mathbf{C}_{i+1}^{\dagger}\mathbf{w} = \mathbf{f}_{i+1} \qquad (A-2)$$

where

$$\mathbf{C}_{i+1} = [\mathbf{C}_i \quad \mathbf{c}_{i+1}] \qquad (A-3)$$

and

$$\mathbf{f}_{i+1} = \begin{bmatrix} \mathbf{f}_i \\ f_{i+1} \end{bmatrix}. \qquad (A-4)$$

To simplify the problem, we only consider the case when the additional constraint vector, $\mathbf{c}_{i+1}$, is of unity length and is orthogonal to all the previous constraint vectors in $\mathbf{C}_i$.

Since the columns of $\mathbf{C}_i$ only span a i-dimensional subspace, it is possible to construct a full rank $N \times (N-i)$ matrix $\mathbf{B}_i$ which is orthogonal to $\mathbf{C}_i$, i.e.,

$$\mathbf{C}_i^{\dagger}\mathbf{B}_i = \mathbf{0}. \qquad (A-5)$$

With the assumption that $\mathbf{c}_{i+1}$ is orthogonal to $\mathbf{C}_i$, we can construct $\mathbf{B}_i$ such that the first column of $\mathbf{B}_i$ is equal to $\mathbf{c}_{i+1}$,

$$\mathbf{B}_i = [\mathbf{c}_{i+1} \ \mathbf{B}_{i+1}]. \qquad (A-6)$$

and $\mathbf{B}_{i+1}$ is orthogonal to $\mathbf{C}_{i+1}$,

$$\mathbf{C}_{i+1}^{\dagger}\mathbf{B}_{i+1} = \mathbf{0}. \qquad (A-7)$$

On the other hand, we can define the matrix $\mathbf{H}$ as the square-root of the covariance matrix $\mathbf{R}_{xx}$,

$$\mathbf{R}_{xx} = \mathbf{H}\mathbf{H}^{\dagger}. \qquad (A-8)$$

Using this $\mathbf{H}$ matrix, we can now define two matrices as

$$\mathbf{C}_{Hi} = \mathbf{H}^{-1}\mathbf{C}_i \qquad (A-9)$$

and

$$\mathbf{B}_{Hi} = \mathbf{H}^{\dagger}\mathbf{B}_i. \qquad (A-10)$$

By direct multiplication, it is easy to verify that

$$\mathbf{C}_{Hi}^{\dagger}\mathbf{B}_{Hi} = \mathbf{0}. \qquad (A-11)$$

The matrix $\mathbf{P}_i$ is defined as the projection matrix of $\mathbf{C}_{Hi}$, which is

$$\mathbf{P}_i = \mathbf{C}_{Hi}(\mathbf{C}_{Hi}^{\dagger}\mathbf{C}_{Hi})^{-1}\mathbf{C}_{Hi}^{\dagger}. \qquad (A-12)$$

Since $\mathbf{B}_{Hi}$ is orthogonal to $\mathbf{C}_{Hi}$, its projection matrix is given by

$$\mathbf{I} - \mathbf{P}_i\ . \qquad (A-13)$$

Several important matrix identities

Lemma 1 *The following two matrix identities are true,*

$$\mathbf{R}_{xx}^{-1}\mathbf{C}_i(\mathbf{C}_i^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{C}_i)^{-1}\mathbf{C}_i^{\dagger} = (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\ , \qquad (A\text{-}14)$$

$$\mathbf{B}_i(\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i)^{-1}\mathbf{B}_i^{\dagger}\mathbf{R}_{xx} = (\mathbf{H}^{\dagger})^{-1}[\mathbf{I} - \mathbf{P}_i]\mathbf{H}^{\dagger} \qquad (A\text{-}15)$$

Proof: See Section 2.4.2.

Lemma 2 *Rank one update of the projection matrix $\mathbf{P}_i$ is given by*

$$\mathbf{P}_{i+1} = \mathbf{P}_i + \mathbf{p}_{i+1}\mathbf{p}_{i+1}^{\dagger} \qquad (A-16)$$

where

$$\mathbf{p}_{i+1} = \sqrt{\gamma_{i+1}}[\mathbf{I} - \mathbf{P}_i]\mathbf{q}_{i+1}, \qquad (A\text{-}17)$$

$$\mathbf{q}_{i+1} = \mathbf{H}^{-1}\mathbf{c}_{i+1}/\sqrt{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}}, \qquad (A\text{-}18)$$

$$\gamma_{i+1} = 1/\mathbf{q}_{i+1}^{\dagger}[\mathbf{I} - \mathbf{P}_i]\mathbf{q}_{i+1}. \qquad (A\text{-}19)$$

Proof: This is a standard formula for rank one update of a projection matrix. See [36] for detailed discussion of projection matrices.

Lemma 3 *The following matrix identity is true,*

$$(\mathbf{H}^{\dagger})^{-1}[\mathbf{I} - \mathbf{P}_i]\mathbf{H}^{-1}\mathbf{c}_{i+1} = \alpha(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{c}_{i+1}, \qquad (A-20)$$

where

$$\alpha = \frac{1}{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1}(\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1})^{-1}\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1}} \qquad (A-21)$$

Proof:

Using Eq.(A-15) in Lemma 1 and the fact that $\mathbf{H}^{-1} = \mathbf{H}^{\dagger}\mathbf{R}_{xx}^{-1}$, we have

$$(\mathbf{H}^{\dagger})^{-1}[\mathbf{I} - \mathbf{P}_i]\mathbf{H}^{-1} = (\mathbf{H}^{\dagger})^{-1}[\mathbf{I} - \mathbf{P}_i]\mathbf{H}^{\dagger}\mathbf{R}_{xx}^{-1} \quad \text{(A-22)}$$

$$= \mathbf{B}_i(\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i)^{-1}\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{R}_{xx}^{-1} \quad \text{(A-23)}$$

$$= \mathbf{B}_i(\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i)^{-1}\mathbf{B}_i^{\dagger}. \quad \text{(A-24)}$$

This suggests that

$$(\mathbf{H}^{\dagger})^{-1}[\mathbf{I} - \mathbf{P}_i]\mathbf{H}^{-1}\mathbf{c}_{i+1} = \mathbf{B}_i(\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i)^{-1}\mathbf{B}_i^{\dagger}\mathbf{c}_{i+1} \quad (\mathrm{A}-25)$$

Using the partition of $\mathbf{B}_i$ as given by Eq.(A-6), the matrix $\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i$ can be partitioned as

$$\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i = \begin{bmatrix} \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1} & \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1} \\ \mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1} & \mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1} \end{bmatrix}. \quad (\mathrm{A}-26)$$

Using the matrix inversion lemma, the inverse of this matrix can also be partitioned into four components as

$$(\mathbf{B}_i^{\dagger}\mathbf{R}_{xx}\mathbf{B}_i)^{-1} = \begin{bmatrix} \alpha & \beta^{\dagger} \\ \beta & \Gamma \end{bmatrix} \quad (\mathrm{A}-27)$$

where

$$\alpha = \frac{1}{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1}(\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1})^{-1}\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1}} \quad \text{(A-28)}$$

$$\beta = -\alpha(\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1})^{-1}\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1} \quad \text{(A-29)}$$

$$\Gamma = (\mathbf{B}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{B}_{i+1} - \frac{\beta\beta^{\dagger}}{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}\mathbf{c}_{i+1}})^{-1}. \quad \text{(A-30)}$$

On the other hand, by using the partition of $\mathbf{B}_i$ in Eq.(A-6) and the fact that $\mathbf{B}_{i+1}$ is orthogonal to $\mathbf{c}_{i+1}$, it is easy to verify that the vector $\mathbf{B}_i^{\dagger}\mathbf{c}_{i+1}$ is given by

$$\mathbf{B}_i^{\dagger}\mathbf{c}_{i+1} = [1, 0, \ldots, 0]^{\dagger}. \quad (\mathrm{A}-31)$$

Multiplying this vector to the matrix inverse given in Eq.(A-27) corresponds to taking its first column, thus Eq.(A-25) becomes

$$(\mathbf{H}^{\dagger})^{-1}[\mathbf{I} - \mathbf{P}_i]\mathbf{H}^{-1}\mathbf{c}_{i+1} = \alpha\mathbf{c}_{i+1} + \mathbf{B}_{i+1}\beta. \quad (\mathrm{A}-32)$$

By directly substituting the definition of β given by Eq.(A-29) into the above equation, we have

$$(\mathbf{H}^\dagger)^{-1}[\mathbf{I}-\mathbf{P}_i]\mathbf{H}^{-1}\mathbf{c}_{i+1} = \alpha\mathbf{c}_{i+1} - \alpha\mathbf{B}_{i+1}(\mathbf{B}_{i+1}^\dagger\mathbf{R}_{xx}\mathbf{B}_{i+1})^{-1}\mathbf{B}_{i+1}^\dagger\mathbf{R}_{xx}\mathbf{c}_{i+1} \tag{A-33}$$

$$= \alpha[\mathbf{I}-\mathbf{B}_{i+1}(\mathbf{B}_{i+1}^\dagger\mathbf{R}_{xx}\mathbf{B}_{i+1})^{-1}\mathbf{B}_{i+1}^\dagger\mathbf{R}_{xx}]\mathbf{c}_{i+1}. \tag{A-34}$$

Using Eq.(A-15) in Lemma 1, the matrix in the bracket of the above equation is equal to

$$(\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger \tag{A-35}$$

and thus the Lemma has been proved.

Lemma 4 *The following matrix identity is true,*

$$(\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger = (\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger + (\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger\mathbf{c}_{i+1}\mathbf{c}_{i+1}^\dagger[\mathbf{I}-(\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger]. \tag{A-36}$$

Proof:
By directly multiplying both sides of Eq.(A-16) by $(\mathbf{H}^\dagger)^{-1}$ on the left and $\mathbf{H}^\dagger$ on the right, we have

$$(\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger = (\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger + (\mathbf{H}^\dagger)^{-1}\mathbf{p}_{i+1}\mathbf{p}_{i+1}^\dagger\mathbf{H}^\dagger. \tag{A-37}$$

By using the definition of $\mathbf{p}_{i+1}$ this equation becomes

$$(\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger = (\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger + \frac{\gamma_{i+1}}{\mathbf{c}_{i+1}^\dagger\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}}(\mathbf{H}^\dagger)^{-1}[\mathbf{I}-\mathbf{P}_i]\mathbf{H}^{-1}\mathbf{c}_{i+1}\cdot \mathbf{c}_{i+1}^\dagger[\mathbf{I}-(\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger]. \tag{A-38}$$

According to Lemma 3, the above equation can be rewritten as

$$(\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger = (\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger + \frac{\gamma_{i+1}\alpha}{\mathbf{c}_{i+1}^\dagger\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}}(\mathbf{H}^\dagger)^{-1}\mathbf{P}_{i+1}\mathbf{H}^\dagger\mathbf{c}_{i+1}\cdot \mathbf{c}_{i+1}^\dagger[\mathbf{I}-(\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger]. \tag{A-39}$$

This Lemma can therefore be proved by showing that

$$\frac{\gamma_{i+1}\alpha}{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}} = 1. \qquad (A-40)$$

From Eq.(A-18) and (A-19), γ_{i+1} can be expressed as

$$\gamma_{i+1} = \frac{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}}{\mathbf{c}_{i+1}^{\dagger}(\mathbf{H}^{\dagger})^{-1}[\mathbf{I}-\mathbf{P}_i]\mathbf{H}^{-1}\mathbf{c}_{i+1}}. \qquad (A-41)$$

Applying Lemma 3 to the denominator, the above equation becomes

$$\gamma_{i+1} = \frac{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}}{\alpha\mathbf{c}_{i+1}^{\dagger}(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{c}_{i+1}} \qquad (A-42)$$

or, equivalently,

$$\frac{\mathbf{c}_{i+1}^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{c}_{i+1}}{\gamma_{i+1}\alpha} = \mathbf{c}_{i+1}^{\dagger}(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{c}_{i+1}. \qquad (A-43)$$

By using Eq.(A-14) in Lemma 1 and the fact that $\mathbf{c}_{i+1}$ is of unity length and is orthogonal to $\mathbf{C}_i$, it is easy to verify that

$$\mathbf{c}_{i+1}^{\dagger}(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{c}_{i+1} = 1 \qquad (A-44)$$

As a result, Eq.(A-40) follows and the Lemma has been proved.

The recursive update formula

Theorem 6 *The optimal weight vector for a linearly-constrained filter with i constraints given by Eq.(A-1) can be expressed as*

$$\mathbf{w}_i = (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\mathbf{w}_{qi} \qquad (A-45)$$

where

$$\mathbf{w}_{qi} = \mathbf{C}_i\mathbf{f}_i. \qquad (A-46)$$

In the case when $\mathbf{f}_i = \mathbf{e}_i$, *the corresponding optimal weight vector is given by*

$$\mathbf{w}_{ci} = (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{-1}\mathbf{c}_i, \qquad (A-47)$$

which is also termed as the primitive weight vector. By adding one more constraint to the filter such that the constraint equations are now given by Eq.(A-2), then the new optimal weight vector can be expressed as

$$\mathbf{w}_{i+1} = \mathbf{w}_i + (f_{i+1} - \mathbf{c}_{i+1}^{\dagger}\mathbf{w}_i)\mathbf{w}_{ci+1} \qquad (\mathrm{A}-48)$$

where $\mathbf{w}_{ci+1}$ *is the primitive weight vector for the filter with* $i+1$ *constraints.*

Proof:

From the closed-form solution in Eq.(2.9), we have

$$\mathbf{w}_i = \mathbf{R}_{xx}^{-1}\mathbf{C}_i(\mathbf{C}_i^{\dagger}\mathbf{R}_{xx}^{-1}\mathbf{C}_i)^{-1}\mathbf{f}_i \qquad (\mathrm{A}-49)$$

By using Eq.(A-14) in Lemma 1, it follows immediately that

$$\mathbf{w}_i = (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\mathbf{w}_{qi}. \qquad (\mathrm{A}-50)$$

When $\mathbf{f}_i = \mathbf{e}_i$ it corresponds to having $\mathbf{w}_{qi} = \mathbf{c}_i$, thus the primitive weight vector, $\mathbf{w}_{ci}$, is given by Eq.(A-47).

Multiplying $\mathbf{w}_{qi+1}$ to both sides of Eq.(A-36) in Lemma 4, we have the following equation,

$$\begin{aligned}(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{w}_{qi+1} &= (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\mathbf{w}_{qi+1} + \\ &\quad (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_{i+1}\mathbf{H}^{\dagger}\mathbf{c}_{i+1}\mathbf{c}_{i+1}^{\dagger}[\mathbf{I} - (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}]\mathbf{w}_{qi+1}\end{aligned} \qquad (\text{A-51})$$

where $\mathbf{w}_{qi+1}$ is defined as

$$\mathbf{w}_{qi+1} = \mathbf{C}_{i+1}\mathbf{f}_{i+1}. \qquad (\mathrm{A}-52)$$

From the partition of $\mathbf{C}_{i+1}$ and $\mathbf{f}_{i+1}$ as given in Eq.(A-3) and Eq.(A-4), $\mathbf{w}_{qi+1}$ can be expressed as

$$\mathbf{w}_{qi+1} = \mathbf{w}_{qi} + f_{i+1}\mathbf{c}_{i+1}. \qquad (\mathrm{A}-53)$$

Thus,

$$(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\mathbf{w}_{qi+1} = (\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\mathbf{w}_{qi} + f_{i+1}(\mathbf{H}^{\dagger})^{-1}\mathbf{P}_i\mathbf{H}^{\dagger}\mathbf{c}_{i+1}. \qquad (\mathrm{A}-54)$$

The second term of the above equation is zero follows from Eq.(A-14) in Lemma 1 and the fact that $\mathbf{c}_{i+1}$ is orthogonal to $\mathbf{C}_i$. Therefore,

$$(\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger\mathbf{w}_{qi+1} = \mathbf{w}_i. \qquad (\mathrm{A}-55)$$

Accordingly, the scale factor in the second term of Eq.(A-51) is given by

$$\mathbf{c}_{i+1}^\dagger[\mathbf{I} - (\mathbf{H}^\dagger)^{-1}\mathbf{P}_i\mathbf{H}^\dagger]\mathbf{w}_{qi+1} = f_{i+1} - \mathbf{c}_{i+1}^\dagger\mathbf{w}_i. \qquad (\mathrm{A}-56)$$

As a result, Eq.(A-51) is in fact equal to Eq.(A-48) and the theorem has been proved.

References

[1] B. Widrow et al. "Adaptive antenna systems". *Proceedings, IEEE*, 55:2143–2139, Dec. 1967.

[2] B. Widrow et al. "Adaptive noise cancelling: Principles and applications". *Proceedings, IEEE*, 63:1692–1716, Dec. 1975.

[3] L. J. Griffiths. "A simple adaptive algorithm for real-time processing in antenna arrays". *Proceedings, IEEE*, 57:1696–1704, Oct. 1969.

[4] N. L. Owsley. "A recent trend in adaptive spatial processing for sensor arrays: constrained adaptation". In J.W.R. Griffiths et al, editor, *Signal processing*, pages 591–604, 1973.

[5] L. J. Griffiths and M. J. Rude. "The P-vector algorithm: a linearly-constrained point of view". In *Proc. of the Asilomar Conference on Signals, Systems and Computers*, Monterey, CA, November 1986.

[6] B. Widrow et al. "Stationary and nonstationary learning characteristics of the LMS adaptive filter". *Proceedings, IEEE*, 64:1151–1162, Aug. 1976.

[7] W.A. Gardner. "Learning characteristics of stochastic-gradient-decent algorithms: A general study, analysis and critique". *Signal Processing*, 6:113–133, 1984.

[8] B. Widrow et al. "The complex LMS algorithm". *Proceedings, IEEE*, 63:719–720, April 1975.

[9] O.L. Frost. "An algorithm for linearly constrained adaptive array processing". *Proceedings, IEEE*, 60:926–935, Aug. 1972.

[10] S. P. Applebaum and D. J. Chapman. "Adaptive arrays with main bean constraints". *IEEE Trans. on Antennas and Propagation*, AP-24:650–662, Sept. 1976.

[11] L.J. Griffiths and C.W. Jim. "An alternative approach to linearly constrained adaptive beamforming". *IEEE Trans. on Antennas and Propagation*, AP-30:27–34, Jan. 1982.

[12] K. M. Buckley and L. J. Griffiths. "An adaptive generalized sidelobe canceller with derivative constraints". *IEEE Trans. on Antennas and Propagation*, AP-34:311–319, Mar. 1986.

[13] K. M. Buckley. "Spatial/spectral filtering with linearly constrained minimum variance beamformers". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-35:249–266, Mar 1987.

[14] B. Widrow and J. M. McCool. "A comparison of adaptive algorithms based on the methods of steepest descent and random search". *IEEE Trans. on Antennas and Propagation*, AP-24:615–637, Sept. 1976.

[15] L. L. Horowitz and K. D. Senne. "Performance advantage of complex LMS for controlling narrow-band adaptive array". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-29:722–736, June 1981.

[16] A. Feuer and E. Weinstein. "Convergence analysis of LMS filters with uncorrelated Gaussian data". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-33:222–229, Feb. 1985.

[17] K. Takao and K. Komiyama. "An adaptive antenna array under directional constraint". *IEEE Trans. on Antennas and Propagation*, AP-24:662–669, Sept. 1976.

[18] M. H. Er and A. Cantoni. "Derivative constraints for braod-band element space antenna array processors,". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-31:1378–1393, Dec 1983.

[19] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1983.

[20] N. K. Jablon. "Steady state analysis of the generalized sidelobe canceller by adaptive noise cancelling techniques". *IEEE Trans. on Antennas and Propagation*, AP-34:330–337, Mar. 1986.

[21] C. Y. Tseng and L. J. Griffiths. "Decomposing the blocking matrix in the generalized sidelobe canceller". In *Proc. of the Asilomar Conference on Signals, Systems and Computers*, pages 574–578, Pacific Grove, CA, Nov 1987.

[22] C. Y. Tseng and L. J. Griffiths. "A systematic procedure for implementing the blocking matrix in decomposed form". In *Proc. of the Asilomar Conference on Signals, Systems and Computers*, pages 808–812, Pacific Grove, CA, Nov 1988.

[23] B. S. Byun and A. F. Gangi. "A constrained-elimination technique for linearly constrained array processing". *IEEE Trans. on Geoscience and Remote Sensing*, GE-30:8–15, Jan. 1981.

[24] J. G. McWhirter. "Recursive least-squares minimization using a systolic array". In Keith Bromley, editor, *Real Time Signal Processing VI, Proceedings. SPIE 431*, pages 105–112, 1983.

[25] C.F.N. Cowan and P.M. Grant. *Adaptive Filters.* Prentice-Hall, 1985.

[26] M. Gentleman. "Least squares computations by givens transformations without square roots". *J. Inst. Math. Appl.*, 12:329–336, 1973.

[27] J. G. McWhirter and T. J. Shepherd. "A systolic array for constrained least-squares problems". In Jeffrey Speiser, editor, *Advanced Algorithms and Architectures for Signal Processing, Proceedings. SPIE 696*, pages 80–87, 1986.

[28] B. Van Veen and R. Roberts. "Systolic arrays for partially adaptive beamforming". In *Proc. of the Asilomar Conference on Signals, Systems and Computers*, pages 584–587, Pacific Grove, CA, Nov 1987.

[29] S. Kalson and K. Yao. "A systolic array for linearly constrained least squares filtering". In *Proc. of the Int'l. Conference on Acoustics, Speech, and Signal Processing*, pages 977–980, Apr 1985.

[30] N. J. Bershad and L. Z. Qu. "On the joint characteristic function of the complex scalar LMS adaptive weights". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-32:1166–1175, December 1984.

[31] N. J. Bershad and L. Z. Qu. "On the probability density function of the LMS adaptive filter weights". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-37:43–56, January 1989.

[32] C. Y. Tseng and L. J. Griffiths. "A systolic power-of-two multiplier structure". In *Proc. of the IEEE Int'l Symposium on Circuits and Systems*, pages 146–149, Portland, OR, May 1989.

[33] L. J. Griffiths and K. M. Buckley. "Quiescent pattern control in linearly constrained adaptive array". *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-35:917–926, July 1987.

[34] B. D. Van Veen. "Eigenstructure based partially adaptive array design". *IEEE Trans. on Antennas and Propagation*, AP-36:357–362, Mar. 1988.

[35] C. Y. Tseng and L. J. Griffiths. "A new approach for reducing perturbation sensitivity in linearly-constrained adaptive arrays". Accepted to the IEEE Asilomar Conference November, 1989.

[36] G. Strang. *Linear Algebra and Its Applications.* Academic Press, 1980.